Some highlights of the system research of the ASCI 30T supercomputer

Fabrizio Petrini,

CCS-3 Modeling, Algorithms, and Informatics Group

Los Alamos National Laboratory

fabrizio@lanl.gov

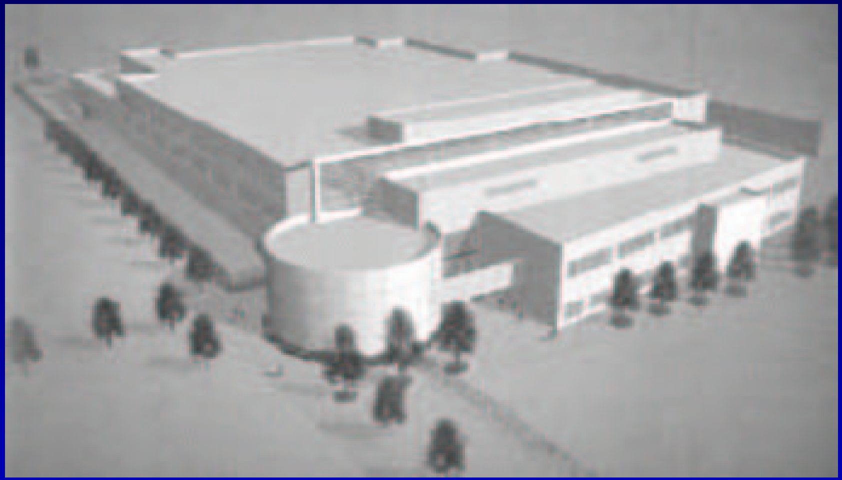
http://www.c3.lanl.gov/~fabrizio

30T Machine at Los Alamos National Laboratory

- Joint project between Compaq and Los Alamos National Laboratory
- 30 TeraOps, expected to be delivered by the end of 2002, #1 in the top 500
- 10 TeraOps delivered by February 5 2002, operational by April 2002
- Total contract price - $215M

Overview of the Facility

- 43,500 sq. ft. unobstructed computer room
- Power 7.1 MW expandable to 30 MW
- Water 130,000 GPD expandable to to 215,000 GPD



Overview of the Facility (continued)

Overview of the Facility (continued)

System Configuration of the Initial System

- 1024 4-processor AlphaServer Es45s
- 4096 Alphas EV68 @ 1Ghz
- 64 nodes with 32GB of memory, 192 with 16 GB, 768 with 8GB ($\simeq$ 11 Terabytes)
- 1312 36-GB hard drives

System configuration of the Initial System

- 2048 Quadrics Elan NICs
- 2 independent network rails with 1024-way federated switches
- 1024 Dual 100 Mbit Ethernet NICs
- 1024 Gbit Ethernet NICs
- 128 RAIDS

and ....

- 1024 True64 Unix
- Alphaserver SC software
- :-(

Research Overview

- Analysis of the network under heavy load and permutation patterns.
- Use of multiple network rails.
- Performance evaluation of I/O traffic and placement of I/O nodes. Interference of background I/O traffic with other jobs.
- *Performance evaluation of the hardware- and software-based collective communication patterns.*
- Job scheduling and resource management (RMS).
- Fault-tolerance of large-scale machines.

Acknowledgments

- Salvador Coll (PhD Student, Politechnical University of Valencia)
- Eitan Fracthenberg (PhD Student, Hebrew University)
- Juan Fernandez Peinador (PhD Student, University of Murcia)
- Adolfy Hoisie (CCS3 Group Leader)
- Wu-chun Feng (external collaborator, CCS-1)

Acknowledgments

- Salvador Coll (PhD Student, Politechnical University of Valencia)
- Eitan Fracthenberg (PhD Student, Hebrew University)
- Juan Fernandez Peinador (PhD Student, University of Murcia)
- Adolfy Hoisie (CCS3 Group Leader)
- Wu-chun Feng (external collaborator, CCS-1)
- Adam Moody (graduate student, Ohio State)

Analysis of the Quadrics network

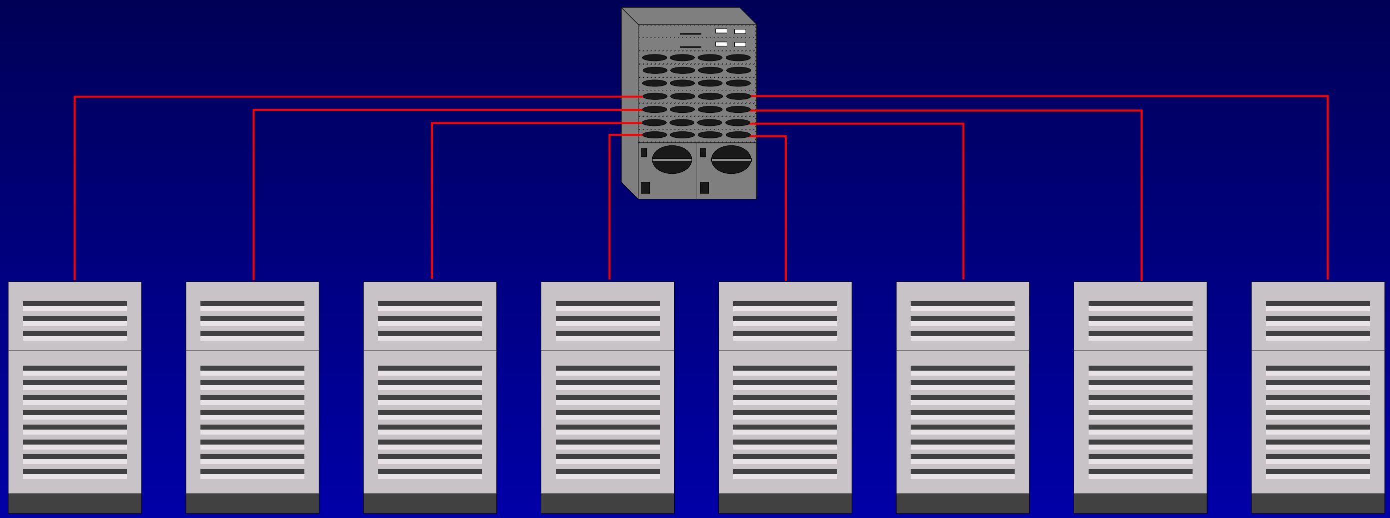
- Fabrizio Petrini, Adolfy Hoisie, Wu-chun Feng and RichardGraham. Performance Evaluation of the Quadrics Interconnection Network, In Workshop on Communication Architecture for Clusters 2001(CAC'01), San Francisco, CA, April 2001.
- Also, Hot Interconnects 2001 and IEEE Micro January-February 2002
- Comprehensive network analysis submitted to Journal of Cluster Computing.

Multiple Independent Network Rails

Using multiple independent networks is an emerging technique to (1) overcome bandwidth limitations and (2) enhance fault-tolerance.

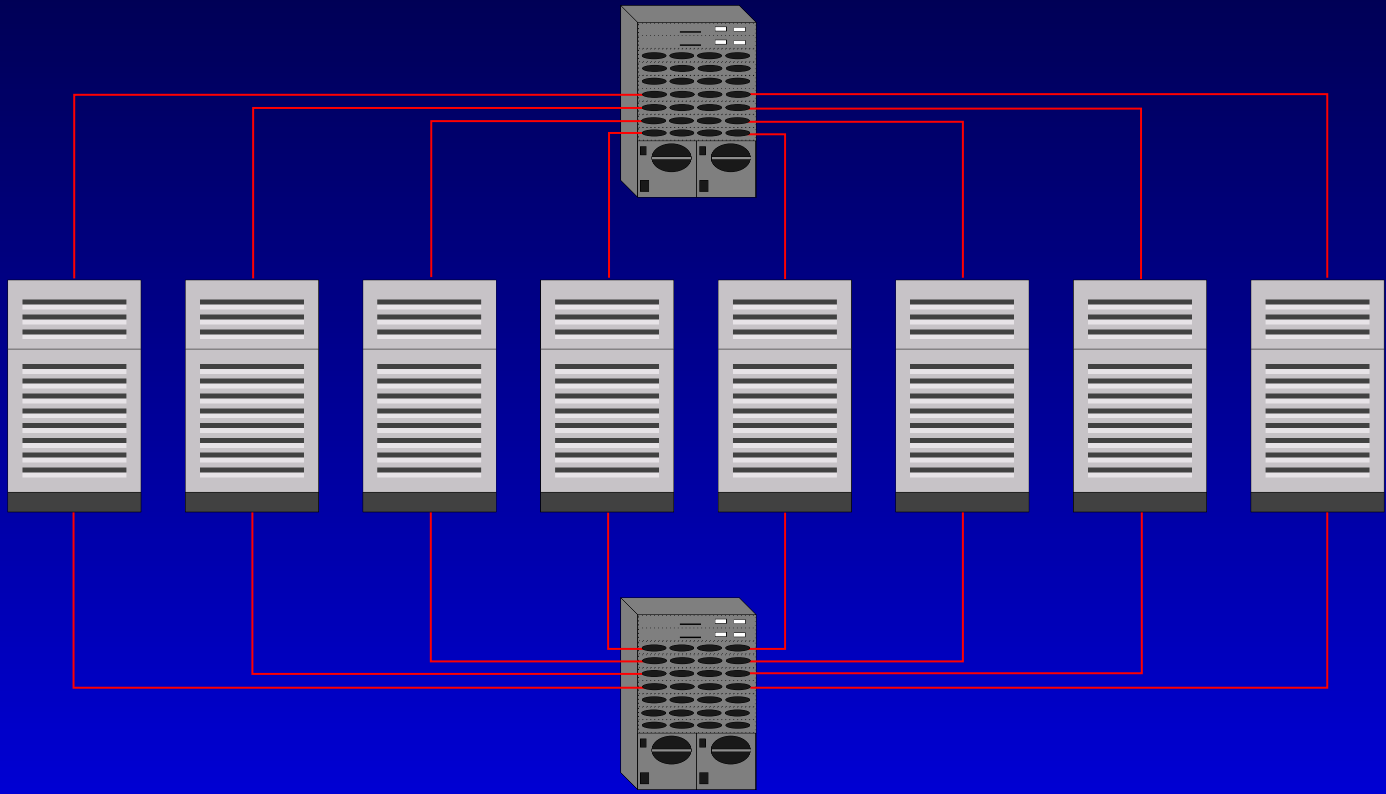
Multiple Independent Network Rails

Using multiple independent networks is an emerging technique to (1) overcome bandwidth limitations and (2) enhance fault-tolerance.



Multiple Independent Network Rails

Using multiple independent networks is an emerging technique to (1) overcome bandwidth limitations and (2) enhance fault-tolerance.



Los Alamos NATIONAL LABORATORY

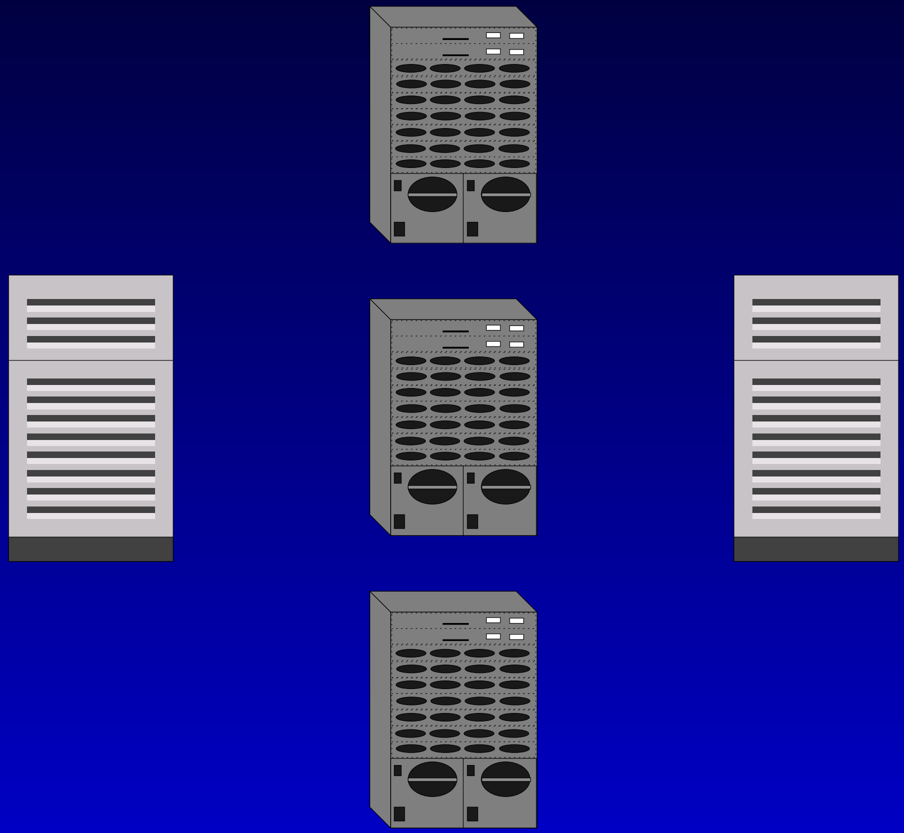
Examples of Multirailed Machines

- ASCI White at Lawrence Livermore National Laboratory – most powerful computer in the world, IBM SP
- The Terascale Computing System (TCS) at the Pittsburgh Supercomputing Center – the second most powerful computer in the world, Quadrics
- ASCI Q machine, currently under development at Los Alamos National Laboratory, Quadrics.
- Infiniband
- Experimental Linux clusters, Quadrics and Myrinet

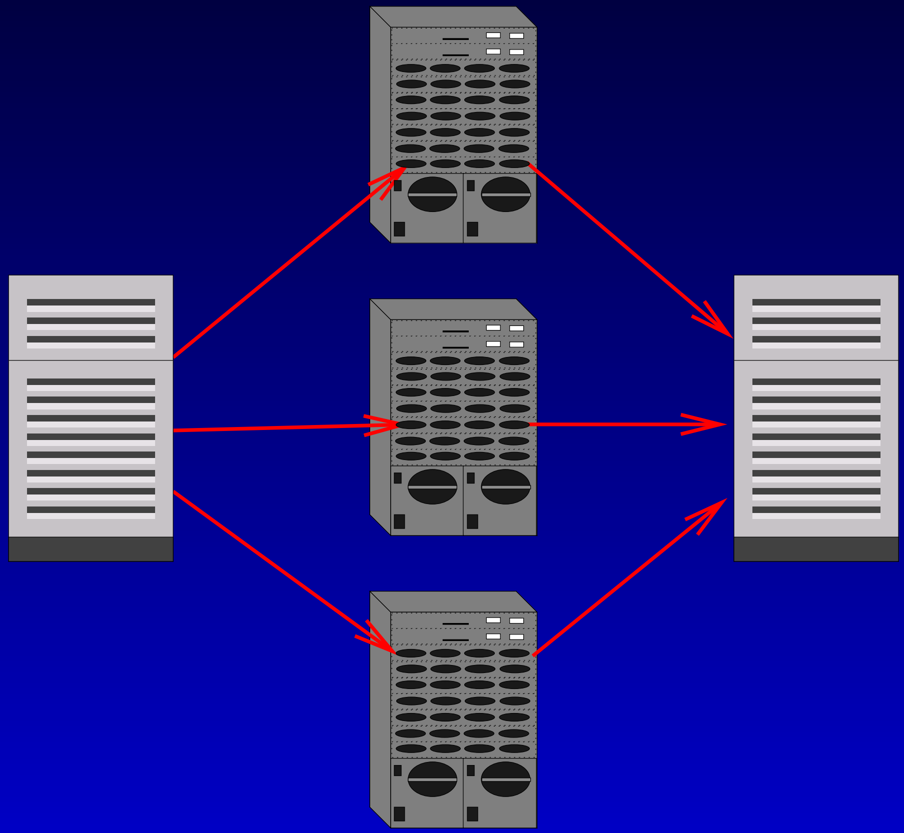
Open Problems

- Rail assignment
- Striping over multiple rails
- Implementation of communication libraries (e.g., MPI, Cray Shmem)
- Multiple rails and I/O interfaces
- Not much is known on how to use multiple rails

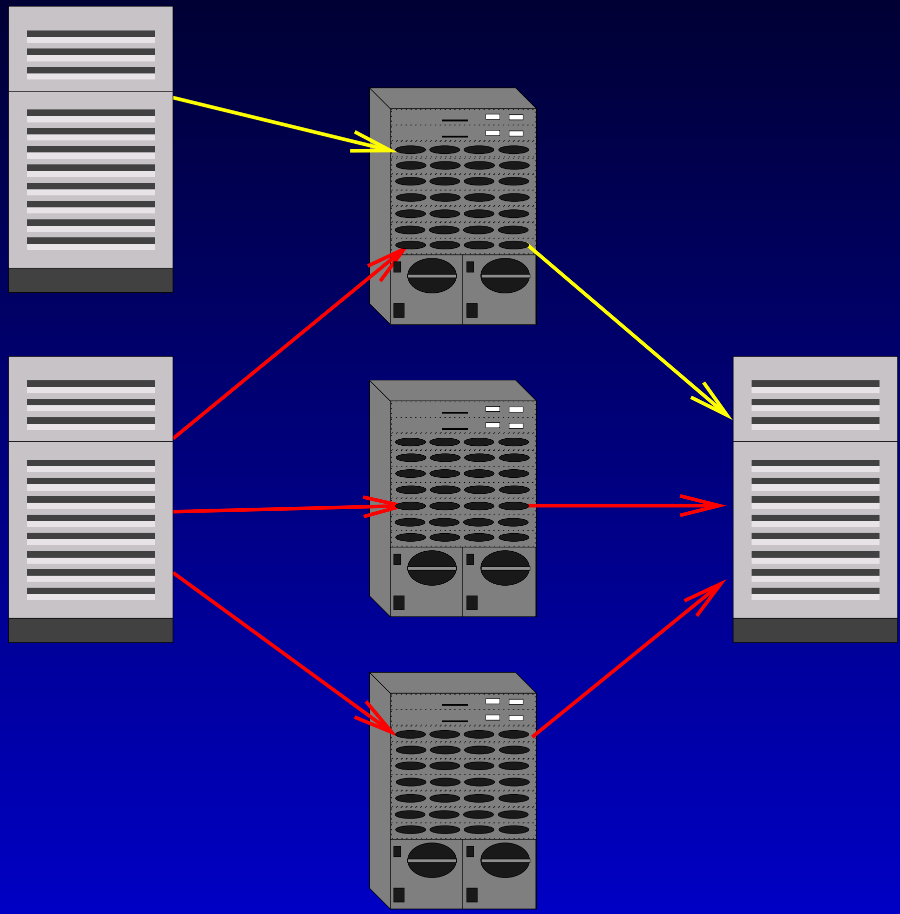
Rail Allocation



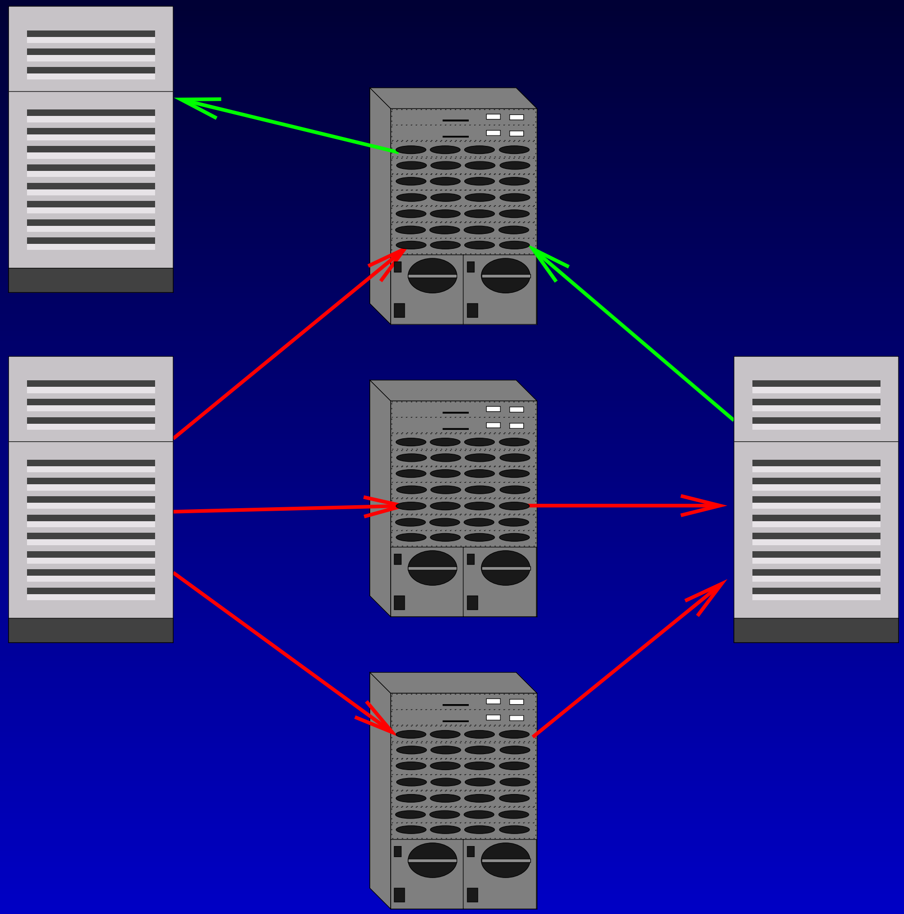
Rail Allocation



Rail Allocation



Rail Allocation



Bidirectional Traffic on the I/O bus

- Most PCI busses cannot efficiently handle bidirectional traffic with high performance networks
- Typically, aggregate bidirectional bandwidth is only 80% of the unidirectional one (Intel 840, Serverworks HE, Compaq Wildfire)
- The same problem is likely to appear in the first Infiniband and PCI-X implementations (e.g., those based on the Intel 870)
- Bidirectional traffic is very common in ASCI applications

State of the Art on Rail Allocation

- A common algorithm to allocate messages to rails is to choose the rail based on the process id of the destination process (rail = destination_id mod RAILS)
- Multiple processes can compete for the same rail even if other rails are available
- No message striping
- No attempt to minimize bidirectional traffic

Outline

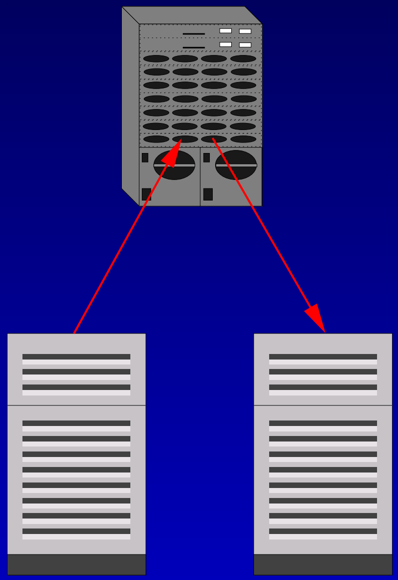
- Basic Algorithm
- Static rail allocation
- Dynamic rail allocation with local information
- Dynamic rail allocation with global information
- Hybrid algorithm
- Experimental evaluation

Basic Algorithm

- The basic algorithm doesn't use any communication protocol
- Whenever a node needs to send a message, it send it on one rail, choosing it in round-robin fashion
- This base case can serve to illustrate the effects of both the overhead of the other protocols and the penalties of the bidirectional traffic

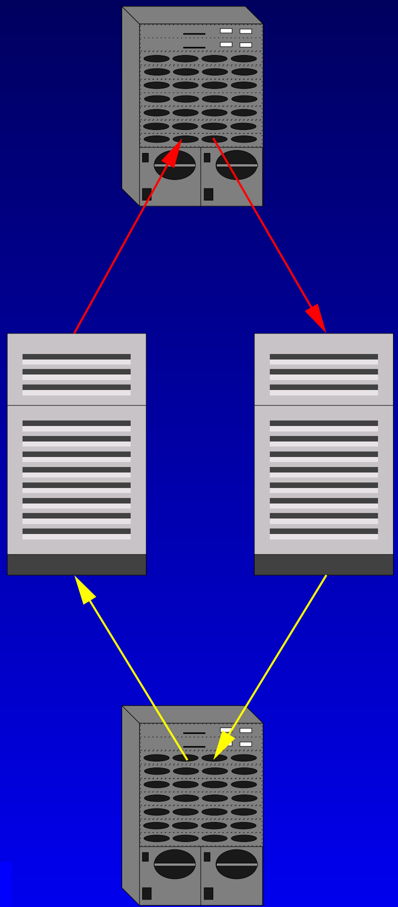
Static Rail Allocation

- With static rail allocation each network interface can either send or receive messages, and the direction is defined at initialization time.



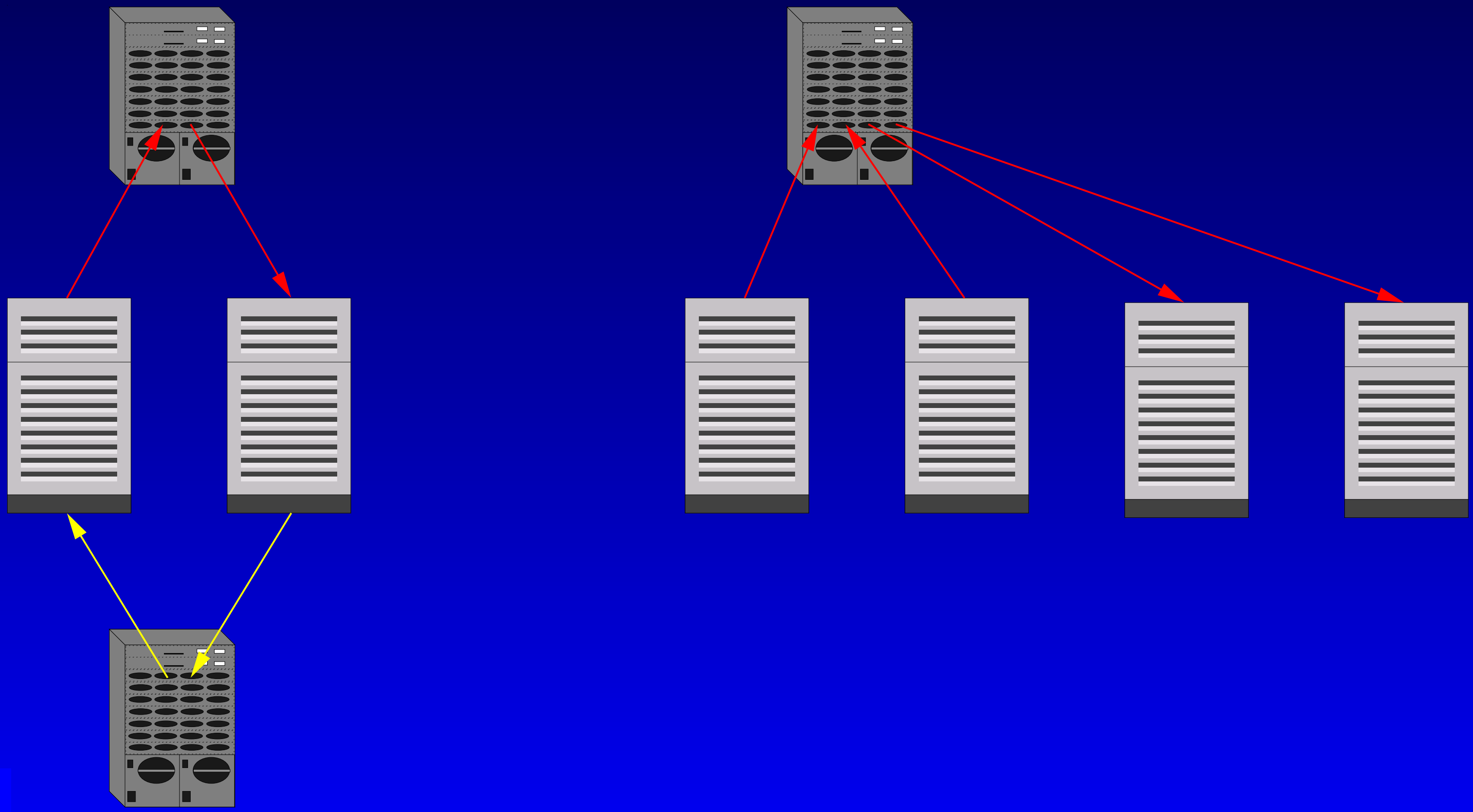
Static Rail Allocation

- With static rail allocation each network interface can either send or receive messages, and the direction is defined at initialization time.



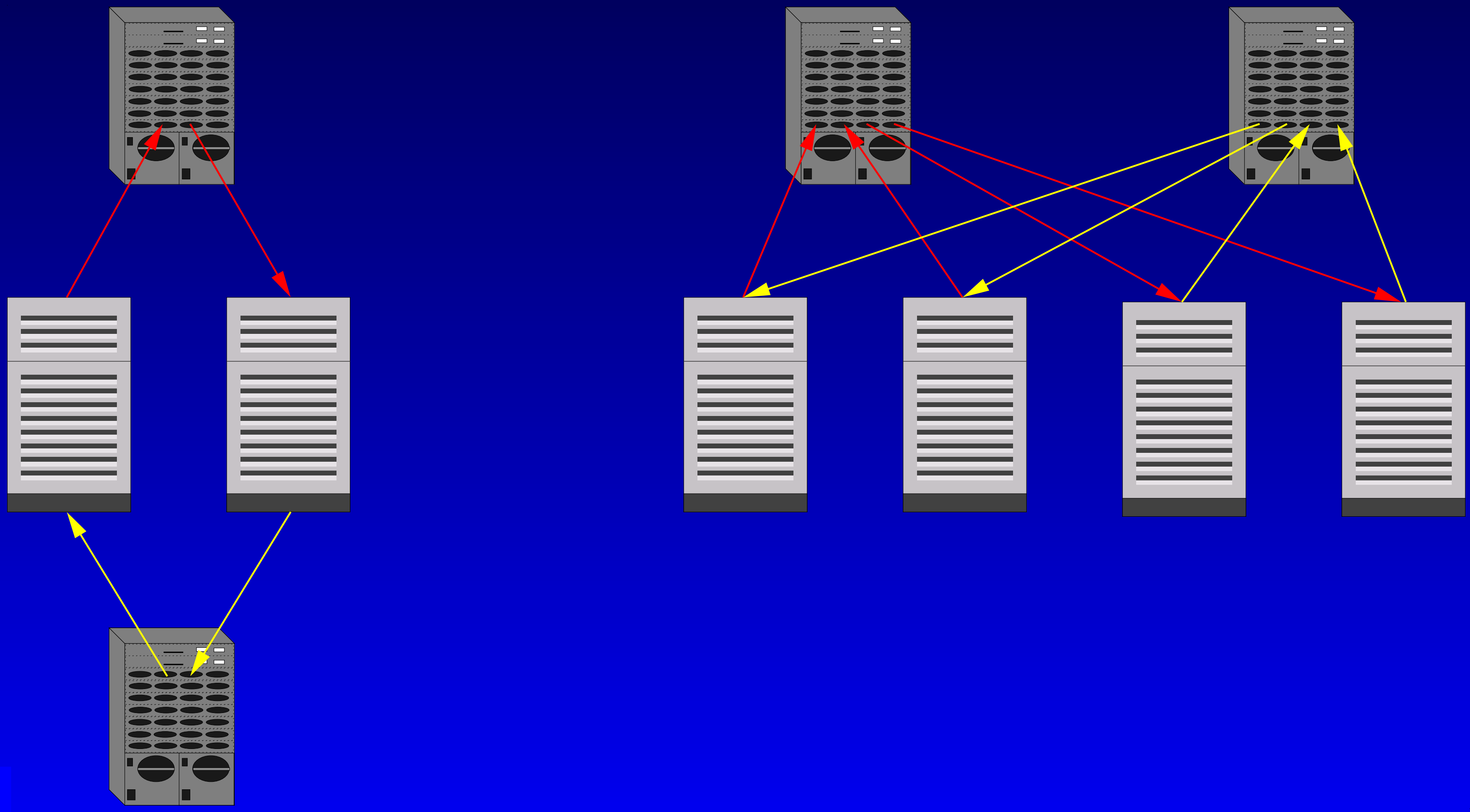
Static Rail Allocation

- With static rail allocation each network interface can either send or receive messages, and the direction is defined at initialization time.



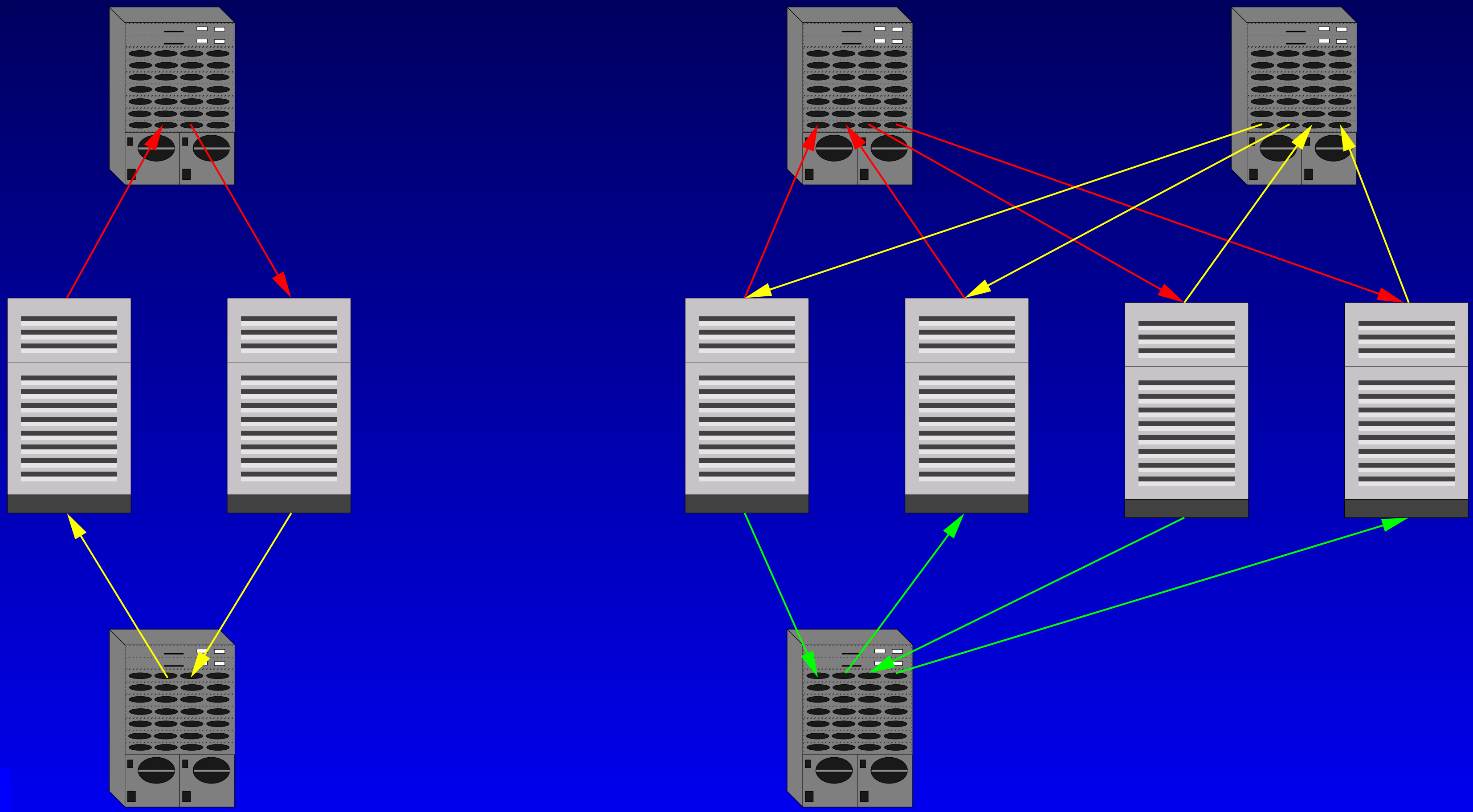
Static Rail Allocation

- With static rail allocation each network interface can either send or receive messages, and the direction is defined at initialization time.



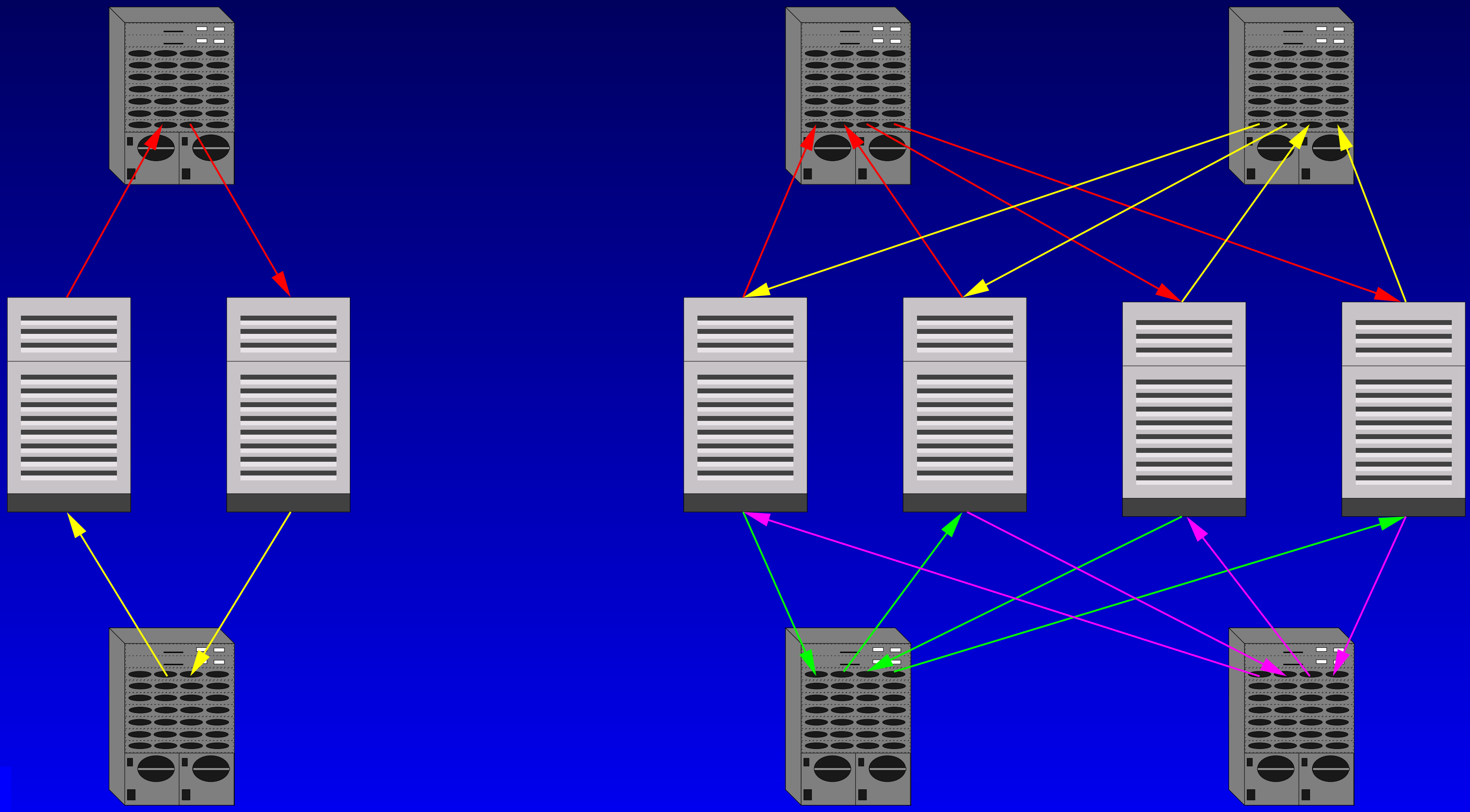
Static Rail Allocation

- With static rail allocation each network interface can either send or receive messages, and the direction is defined at initialization time.



Static Rail Allocation

- With static rail allocation each network interface can either send or receive messages, and the direction is defined at initialization time.



Los Alamos NATIONAL LABORATORY

Lower Bound with Static Rail Allocation

A high number of rails is required for statically allocated unidirectional traffic.

A network with r rails can support no more than n nodes, where

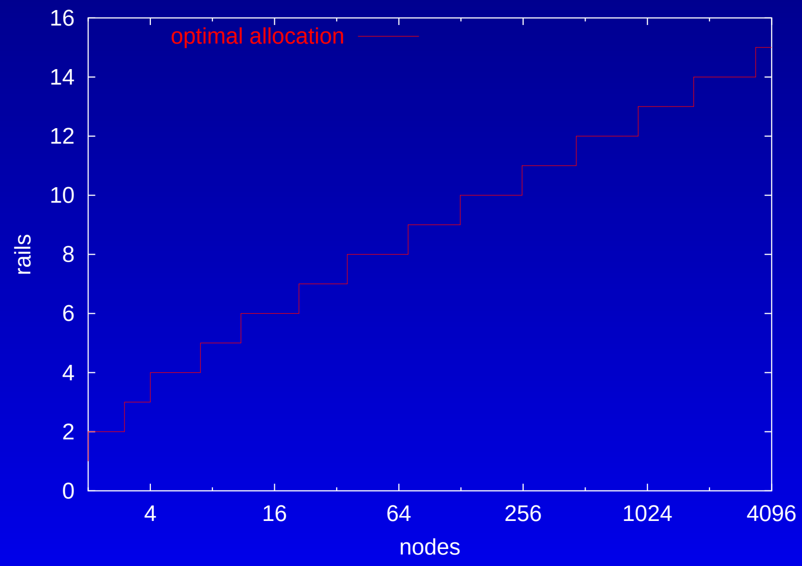
$$n \leq \binom{r}{\left\lfloor \frac{r}{2} \right\rfloor}$$

Los Alamos NATIONAL LABORATORY

Lower Bound with Static Rail Allocation

A high number of rails is required for statically allocated unidirectional traffic.
A network with *r* rails can support no more than *n* nodes, where

$$n \leq \binom{r}{\left\lfloor \frac{r}{2} \right\rfloor}$$



Dynamic Algorithm with Local Information

- With the dynamic algorithms the direction in which each network interface is used can change over time
- The *local-dynamic* algorithm allocates the rails in both directions, using local information available on the sender side
- Messages are sent over rails that not sending or receiving other messages
- Messages can be striped over multiple rails
- There is no guarantee that traffic will be unidirectional

Los Alamos NATIONAL LABORATORY

Dynamic Algorithm with Global Information

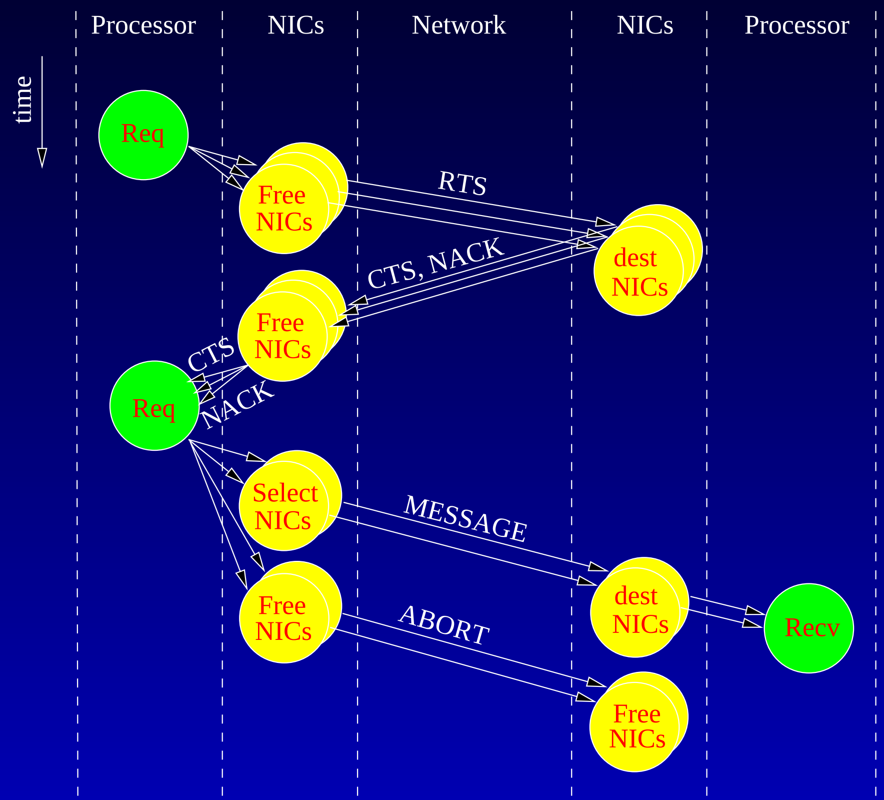
- The *dynamic* algorithm tries to reserve both end-points before sending a message
- In its core there is a sophisticated distributed algorithm that (1) ensures unidirectional traffic at both ends and (2) avoids deadlocks, potentially generated by multiple requests with a cyclic dependency

Los Alamos
NATIONAL LABORATORY

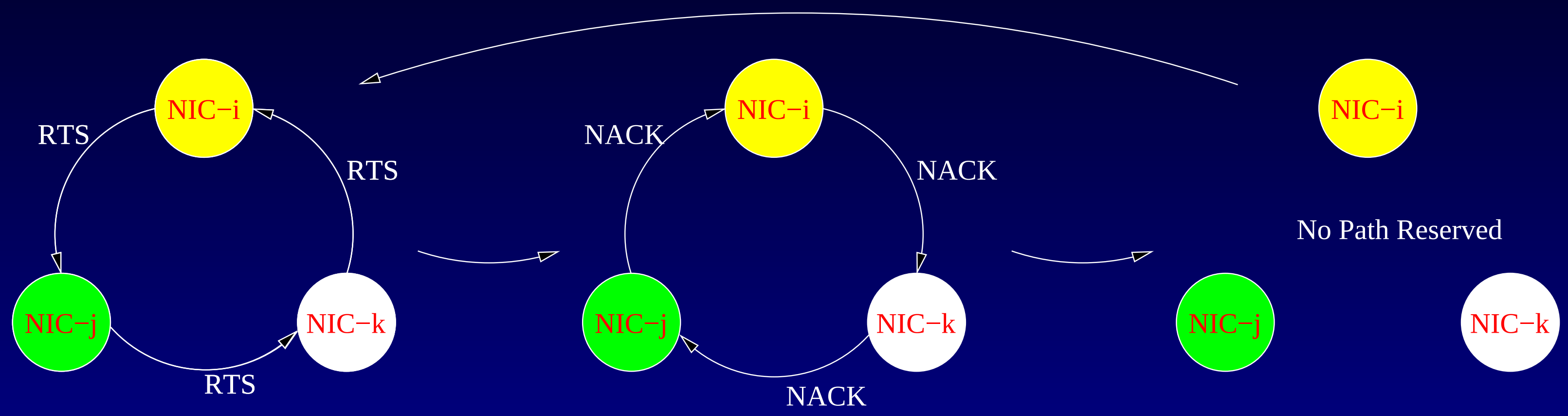
Dynamic Algorithm: Implementation Issues

- The efficient implementation of this algorithm requires some processing power in the network interface, which needs to process control packets and perform the reservation protocol without interfering with the host
- For example, the Quadrics network interface is equipped with a thread processor that can process an incoming packet, do some basic processing and send a reply in as few as $2\mu s$

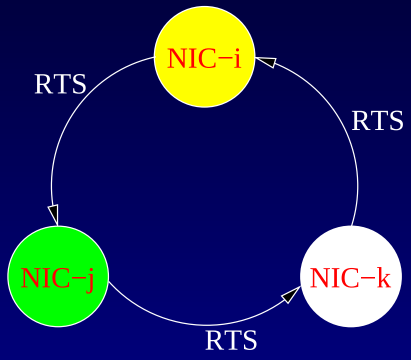
Dynamic Algorithm



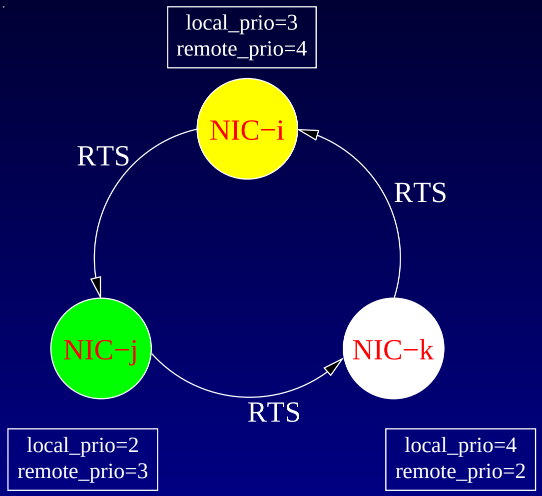
Livelock in the Dynamic Algorithm



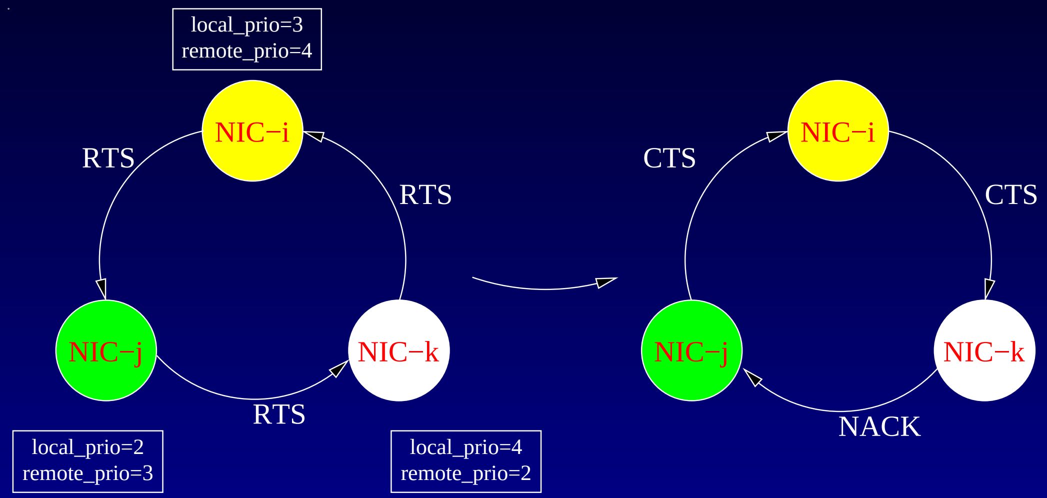
Livelock Avoidance in the Dynamic Algorithm



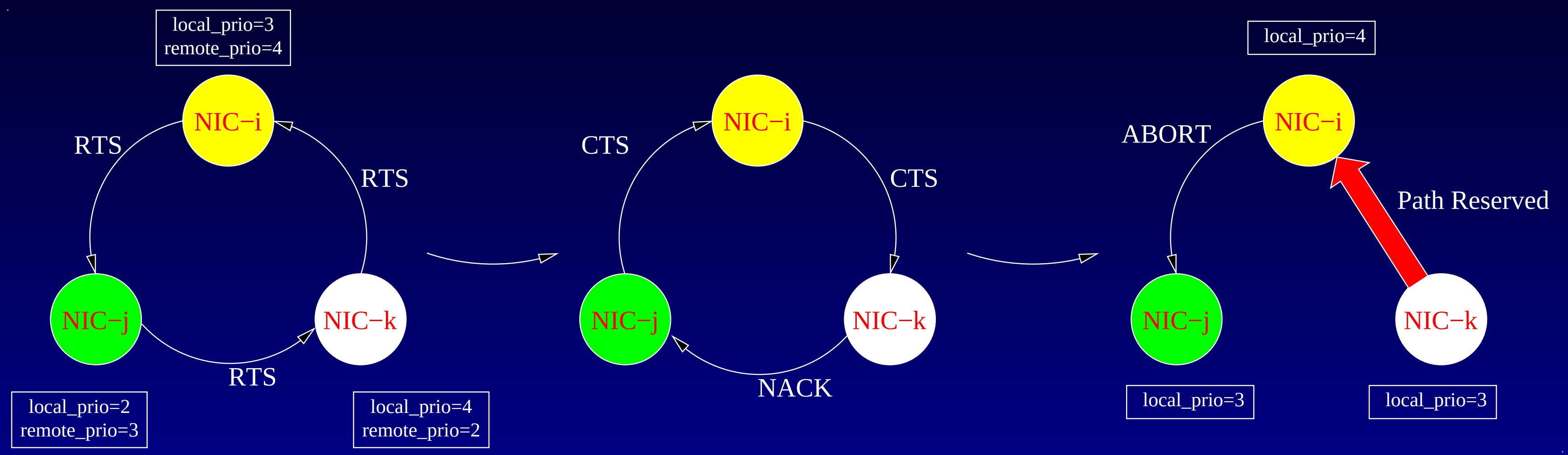
Livelock Avoidance in the Dynamic Algorithm



Livelock Avoidance in the Dynamic Algorithm



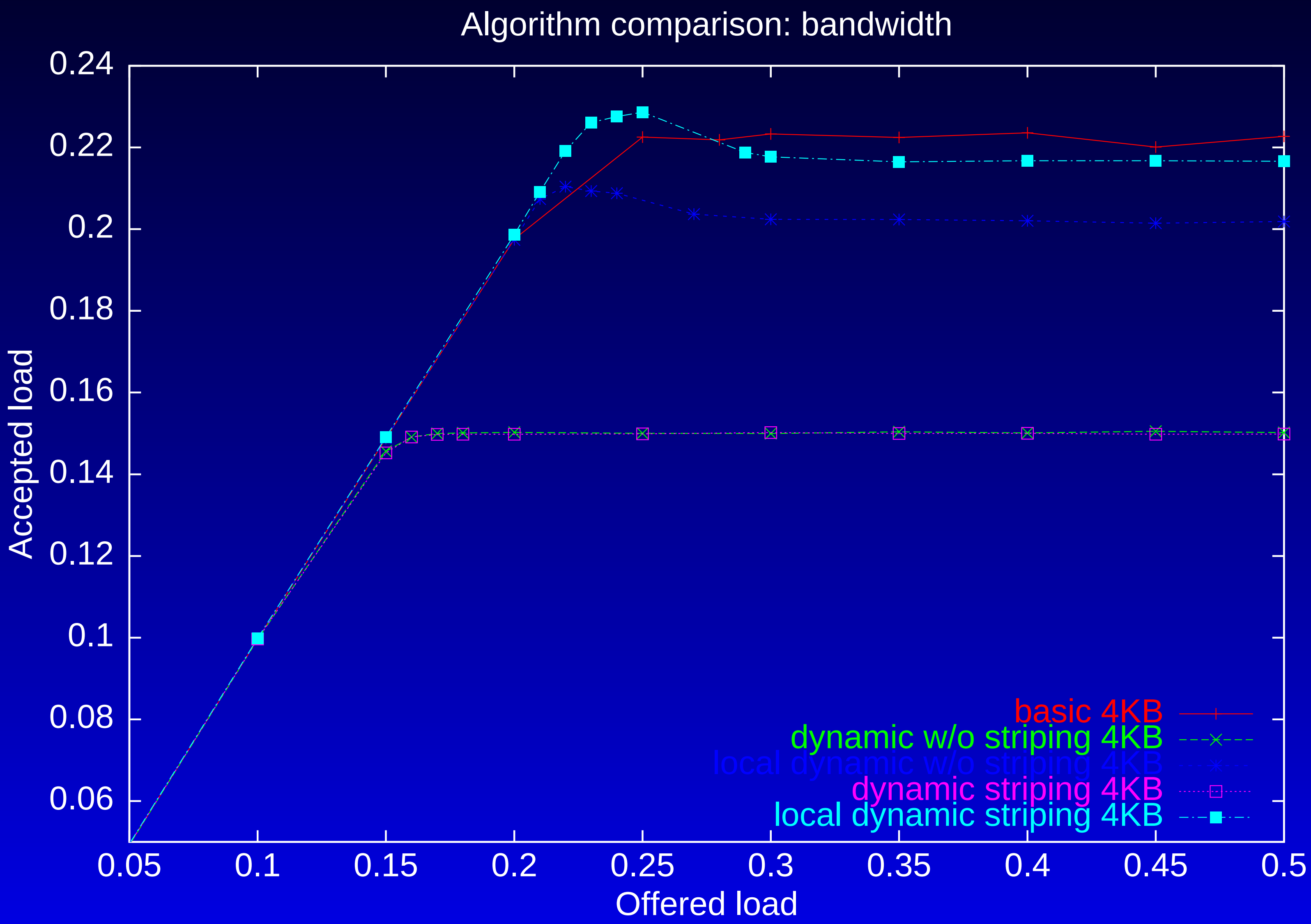
Livelock Avoidance in the Dynamic Algorithm



Hybrid Algorithm

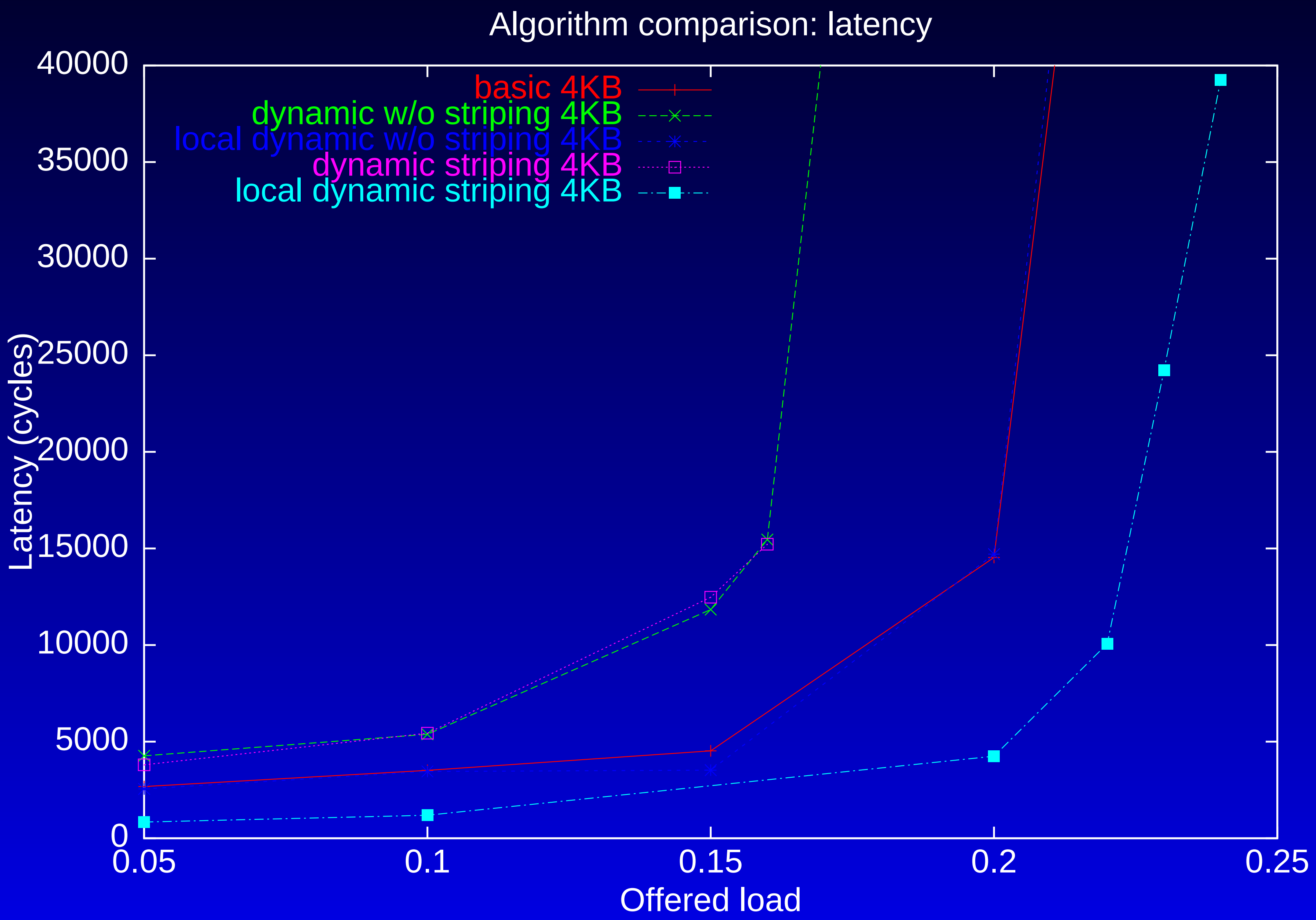
- The dynamic algorithm incurs a substantial overhead, for every message size.
- The hybrid algorithm sends short message without a reservation protocol
- Short messages are not striped
- It can cause bidirectional traffic for a short time

Results - Bandwidth, 4KB messages



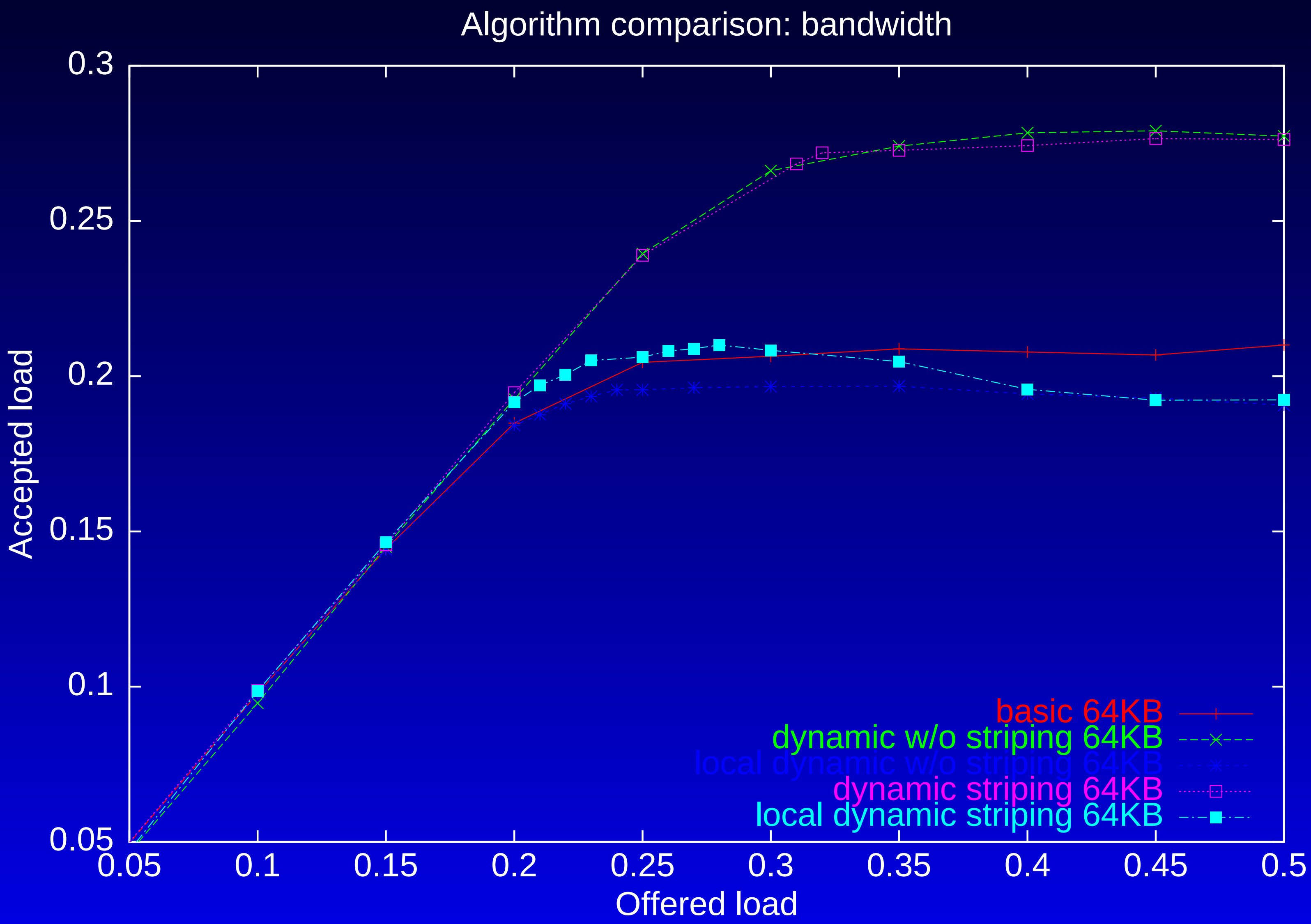
Los Alamos NATIONAL LABORATORY

Results - Latency, 4KB messages

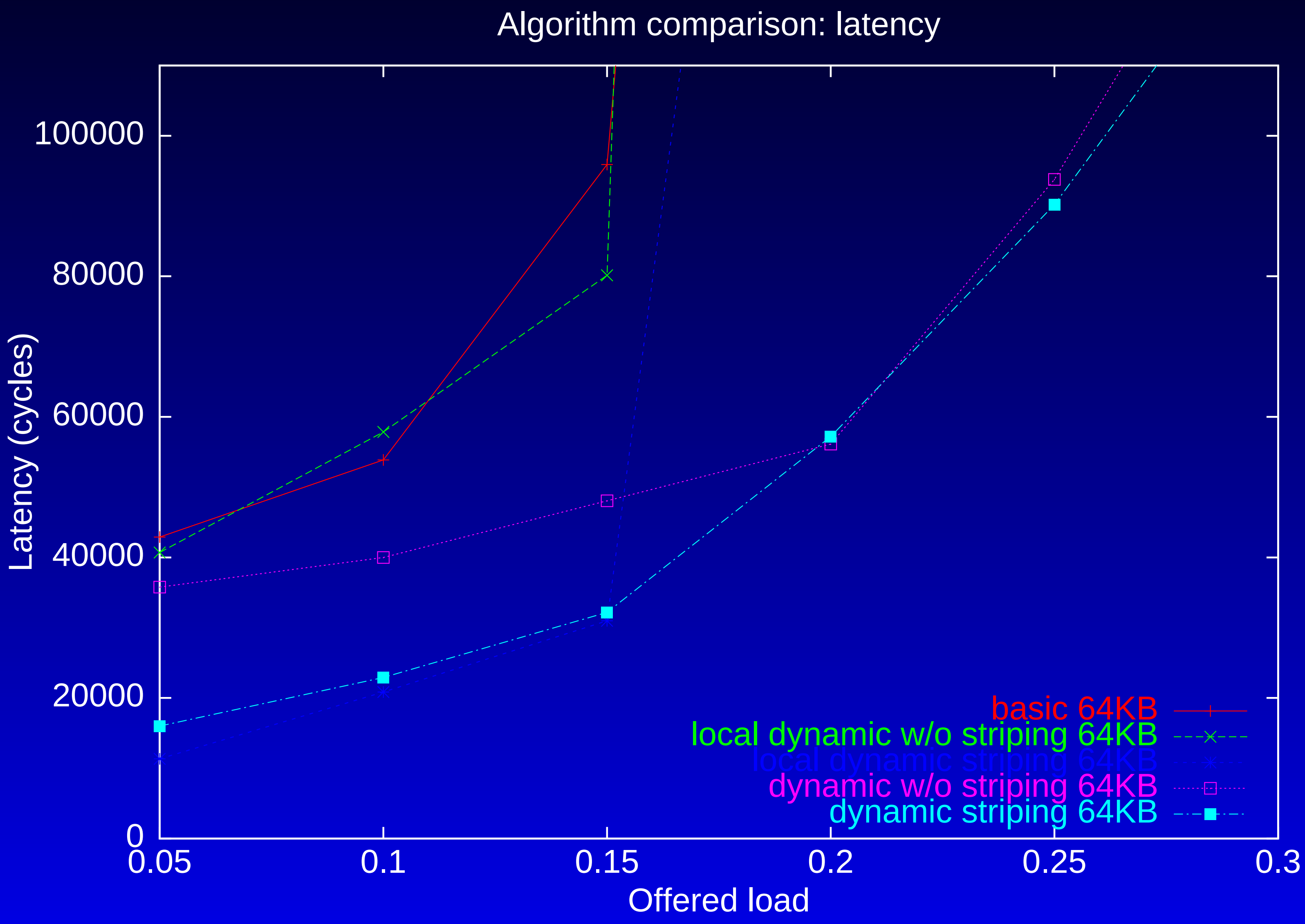


Los Alamos NATIONAL LABORATORY

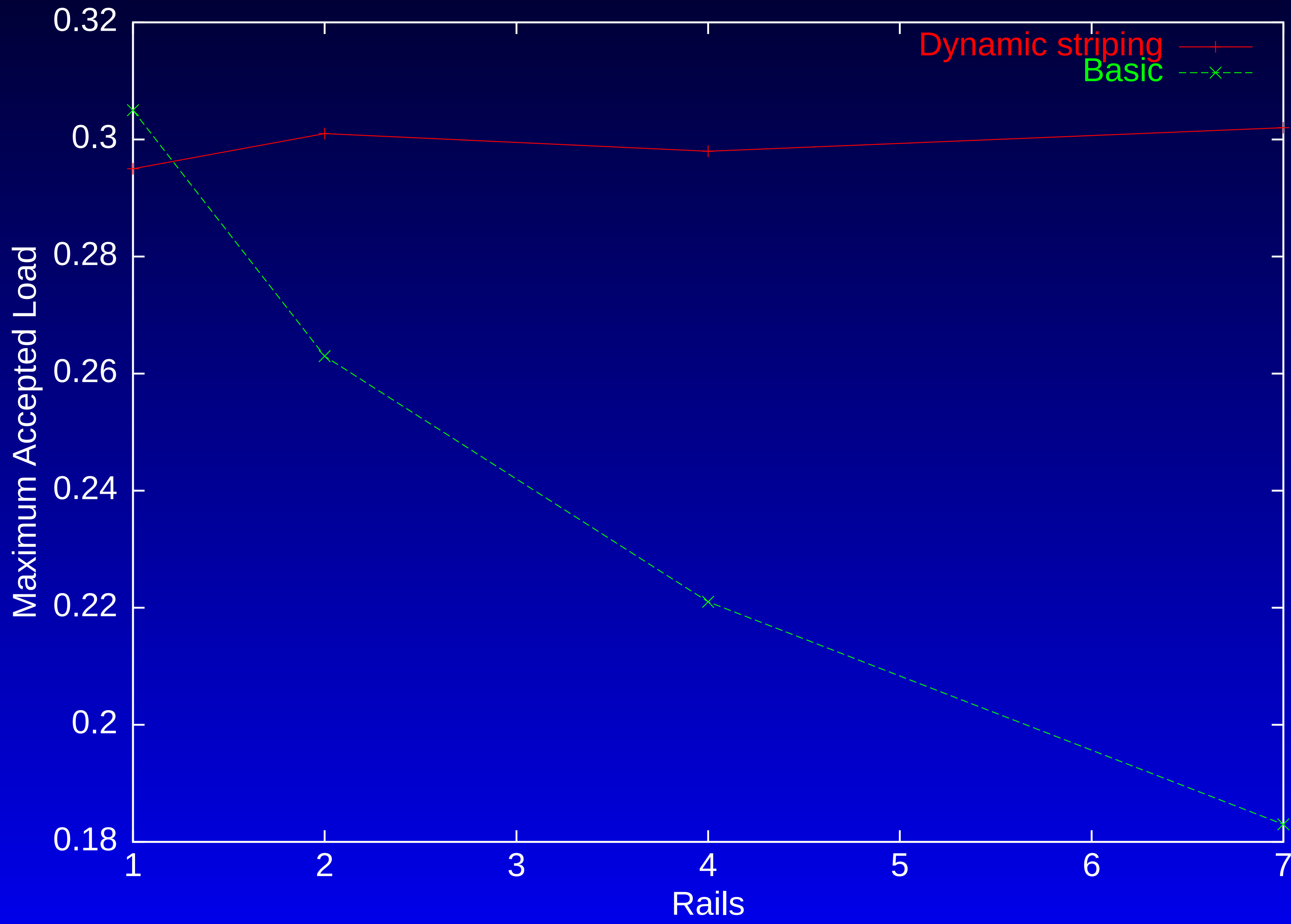
Results - Bandwidth, 64KB messages



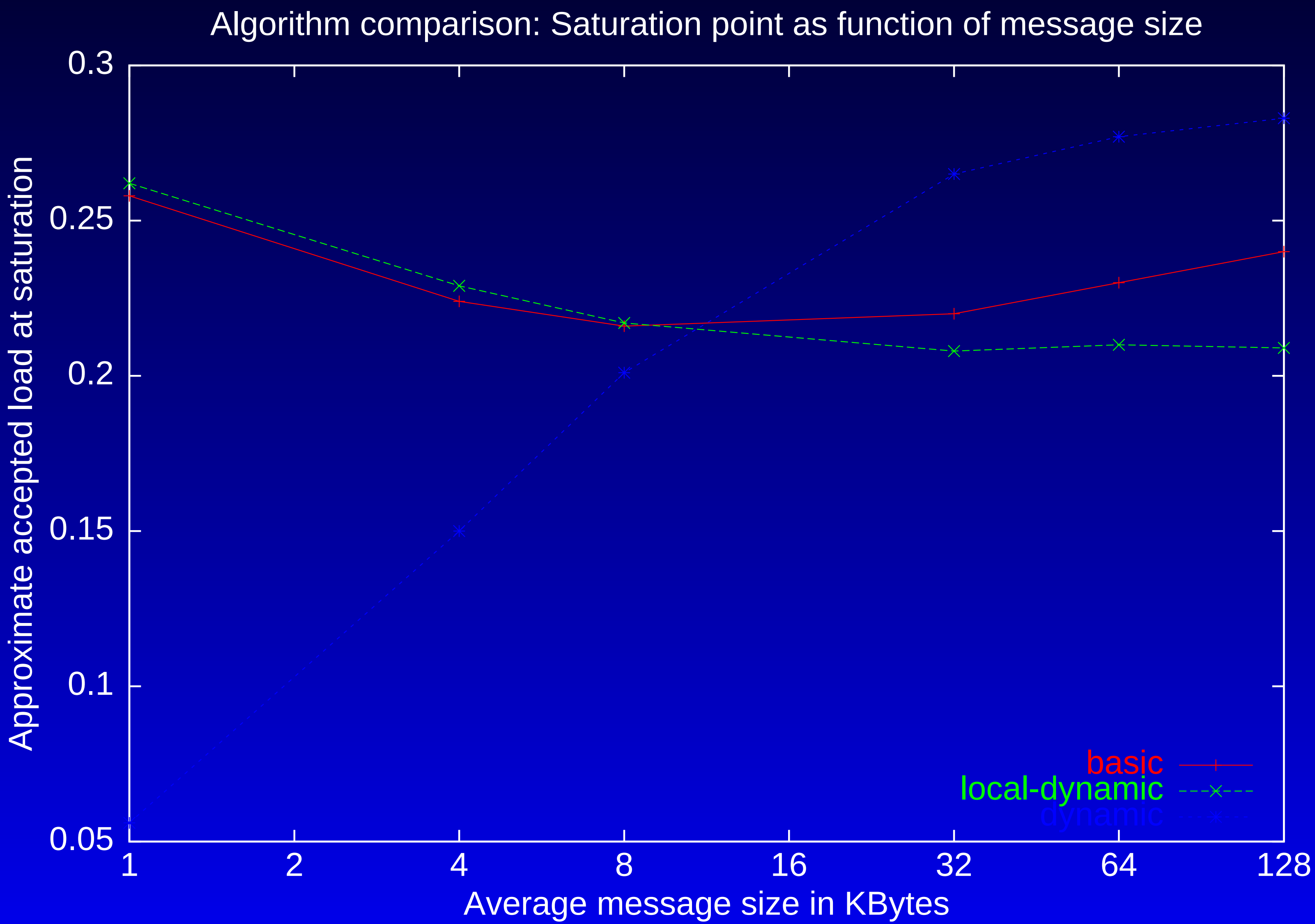
Results - Latency, 64KB messages



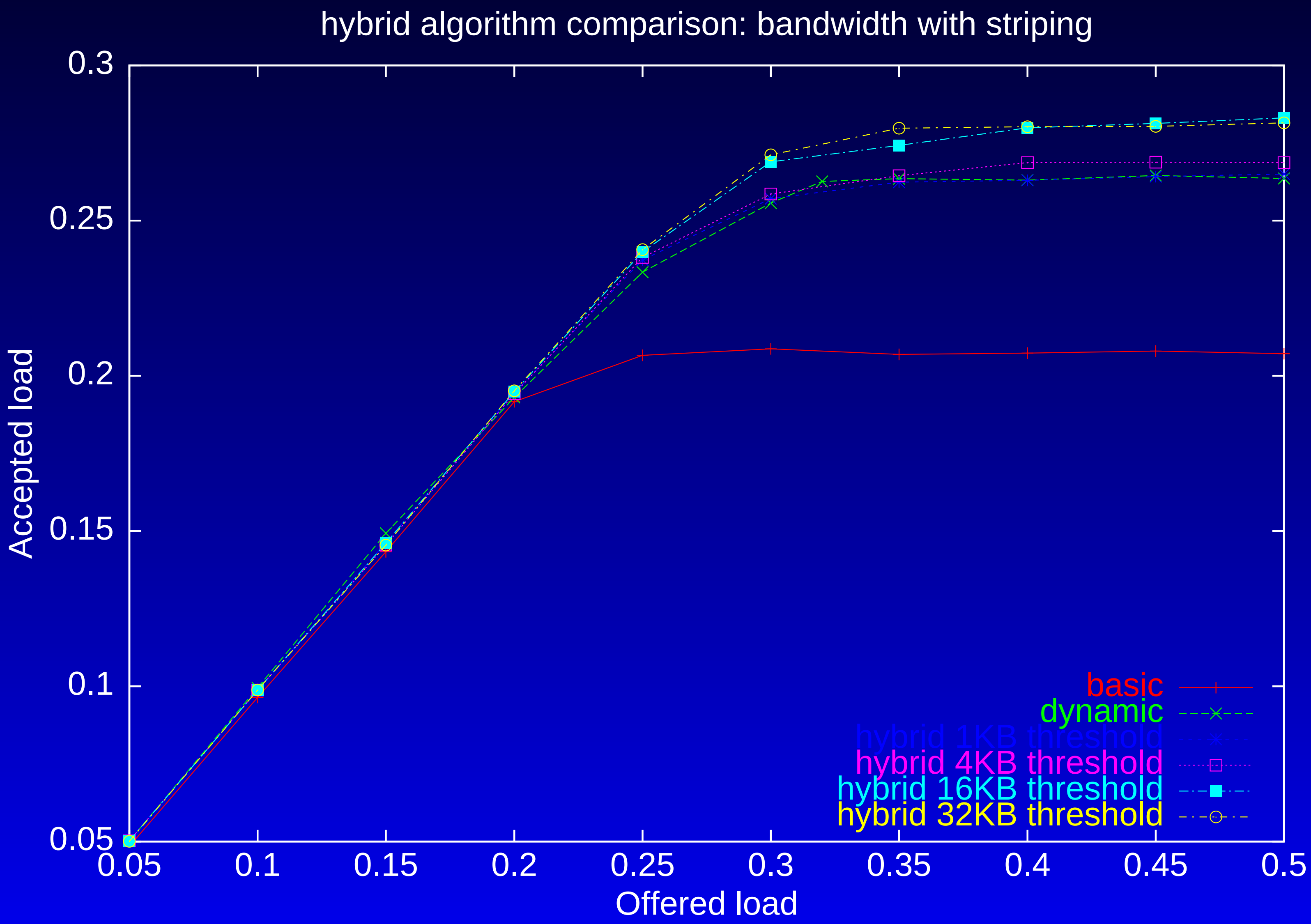
Results - Bandwidth vs Number of Rails



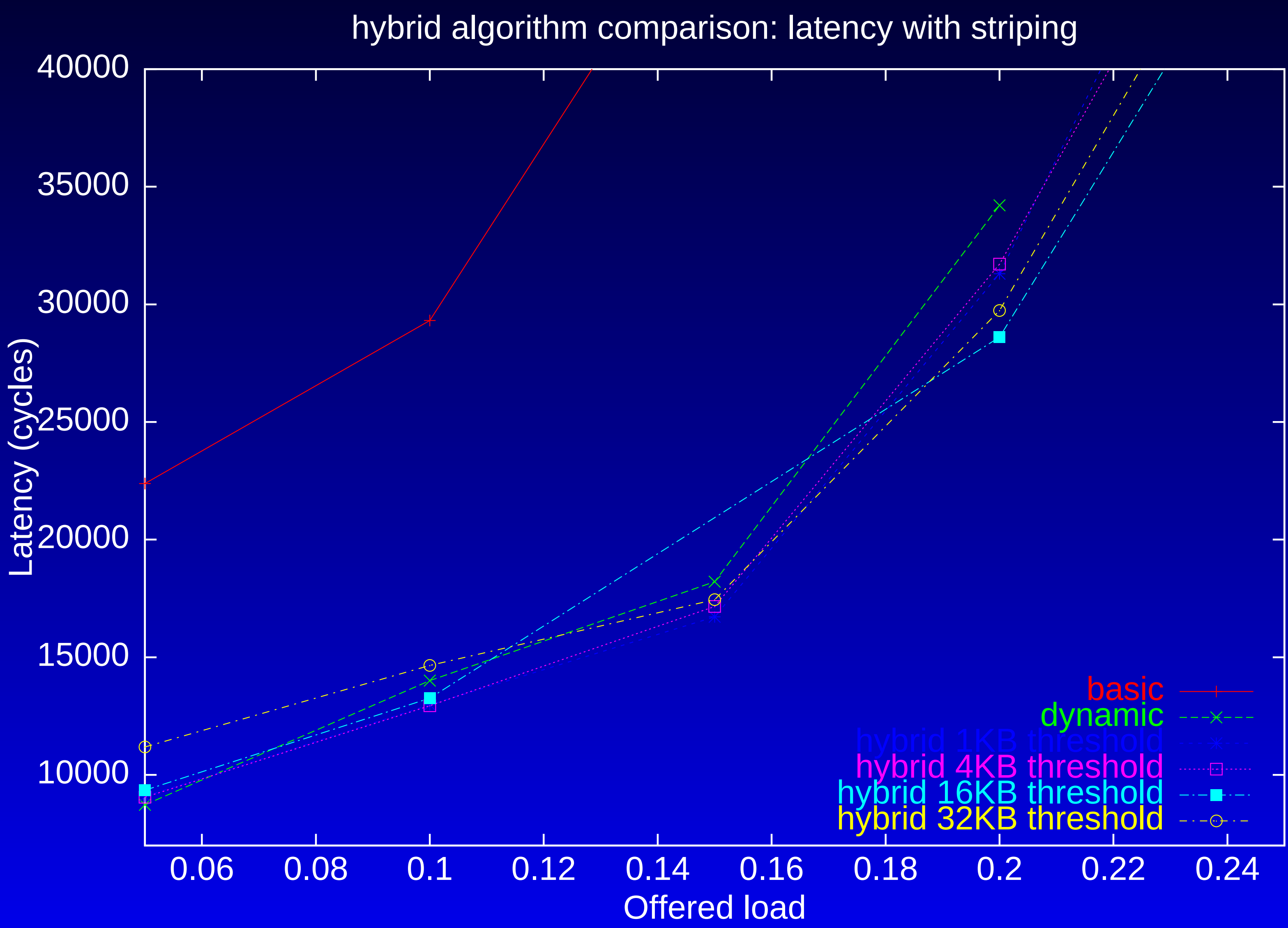
Results - Saturation Points vs Message Size



Results - Hybrid, Bandwidth with Striping



Results - Hybrid, Latency with Striping



Publications on Multirail Algorithms

- Salvador Coll, Eitan Frachtenberg, Fabrizio Petrini, Adolfy Hoisie, and Leonid Gurvits. Using Multirail Networks in High-Performance Clusters. In IEEE Cluster 2001, Newport Beach, CA, October 2001.
- Selected for publication on the journal “Concurrency, Practice and Experience”.

Los Alamos NATIONAL LABORATORY

Open Problems with I/O network traffic

- Characterization of I/O traffic (Time and Space distribution)
- I/O reads and writes
- Placement of I/O nodes
- Running computational tasks on I/O nodes
- Interference between jobs performing I/O and other jobs

Los Alamos
NATIONAL LABORATORY

Performance analysis with a single parallel job

We address five distinct performance dimensions.

- I/O read/write ratio.
- The inter-arrival time between two I/O messages issued by a client node can be either uniformly or exponentially distributed.
- I/O traffic: this parameter defines the access pattern to the I/O nodes. Two patterns have been analyzed:
  - uniform I/O, each node performing I/O randomly selects its destination for every transaction and
  - fixed I/O, each node uses a fixed destination for all its transactions.

Los Alamos NATIONAL LABORATORY

Performance analysis with a single parallel job

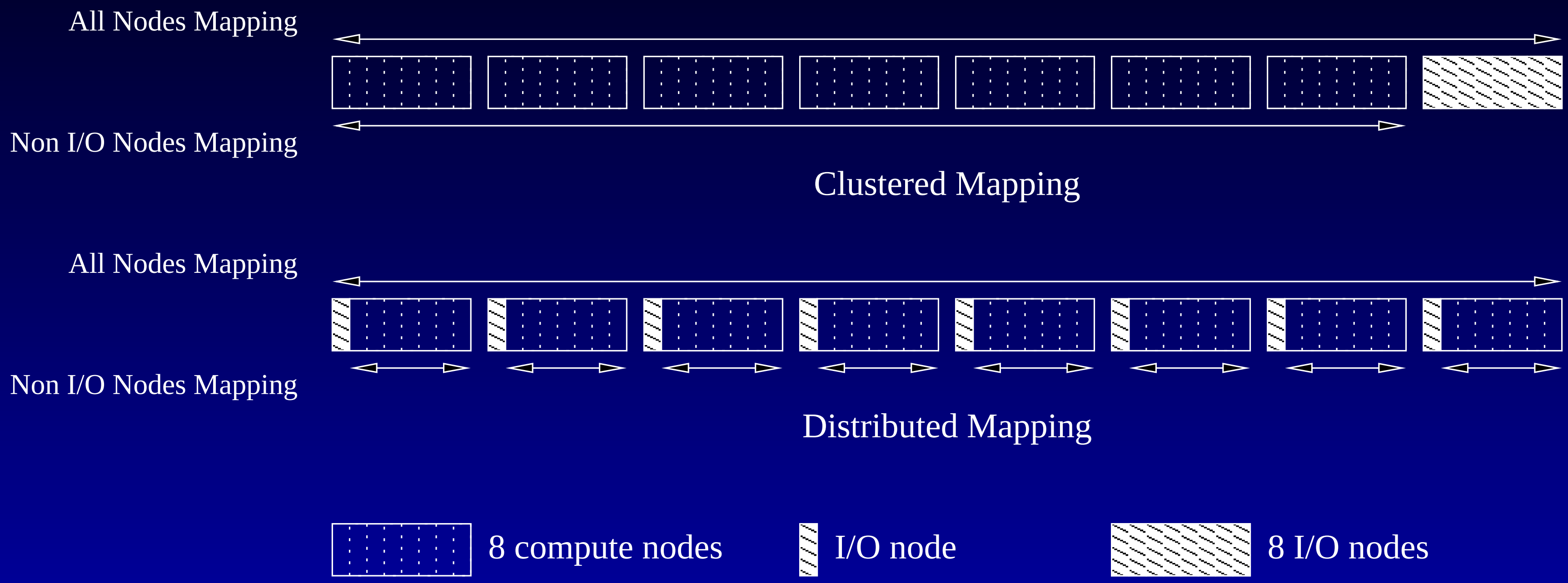
- I/O node mapping (hot-node mapping): this parameter defines the placement of the I/O nodes in the cluster. Two alternatives have been tested:
  - clustered, I/O nodes located in consecutive nodes at the higher nodes locations and
  - distributed, I/O nodes uniformly distributed through the cluster.
- Application mapping: defines whether the application performing I/O runs on the I/O nodes (all nodes mapping) or not (non-I/O nodes mapping).

Los Alamos NATIONAL LABORATORY

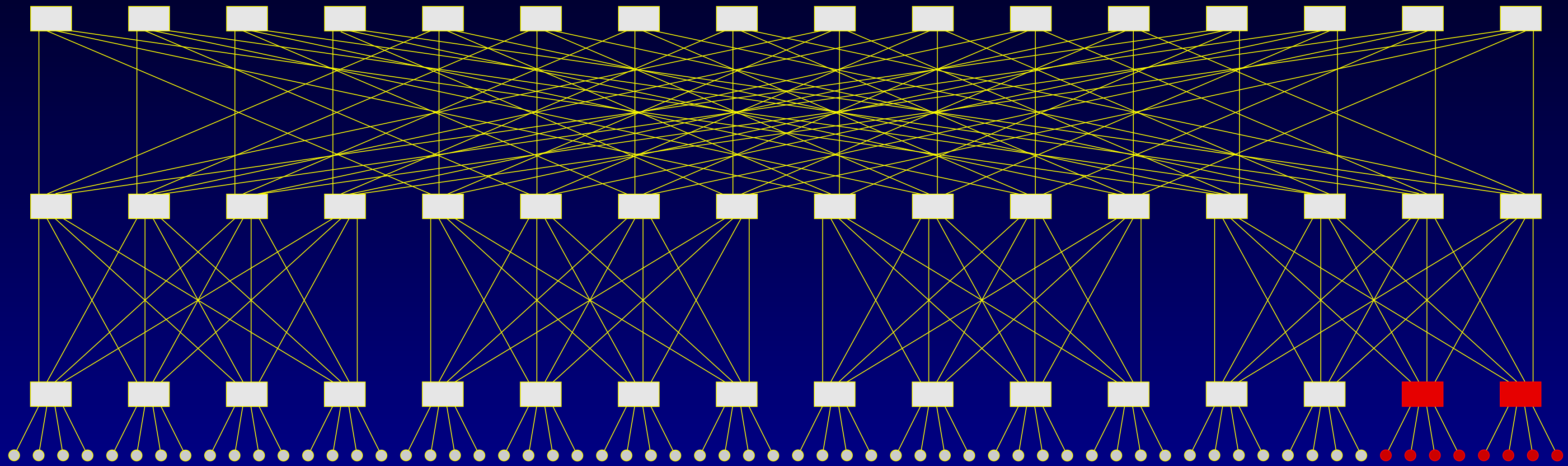
Modeling I/O traffic: multiple hot-spots

- The network traffic generated by a parallel job that is performing input/output can be modeled with a collection of hot-spots, where each hot-spot is a node that acts as an I/O server and is the target of multiple messages originated by the other nodes.
- This test has been designed to analyze the behavior of the network when one parallel job is performing transactions over several hot nodes.

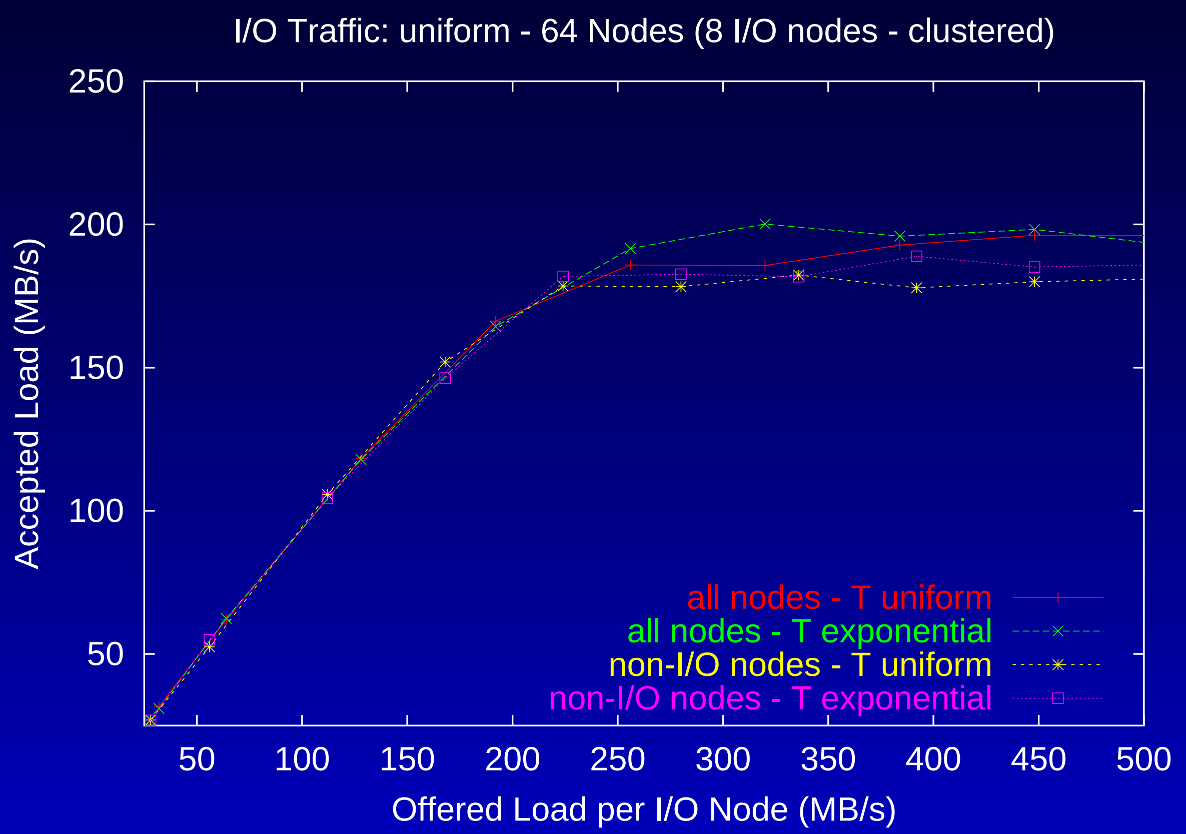
Placement of I/O Nodes in a single parallel job



Placement of I/O Nodes in a single parallel job

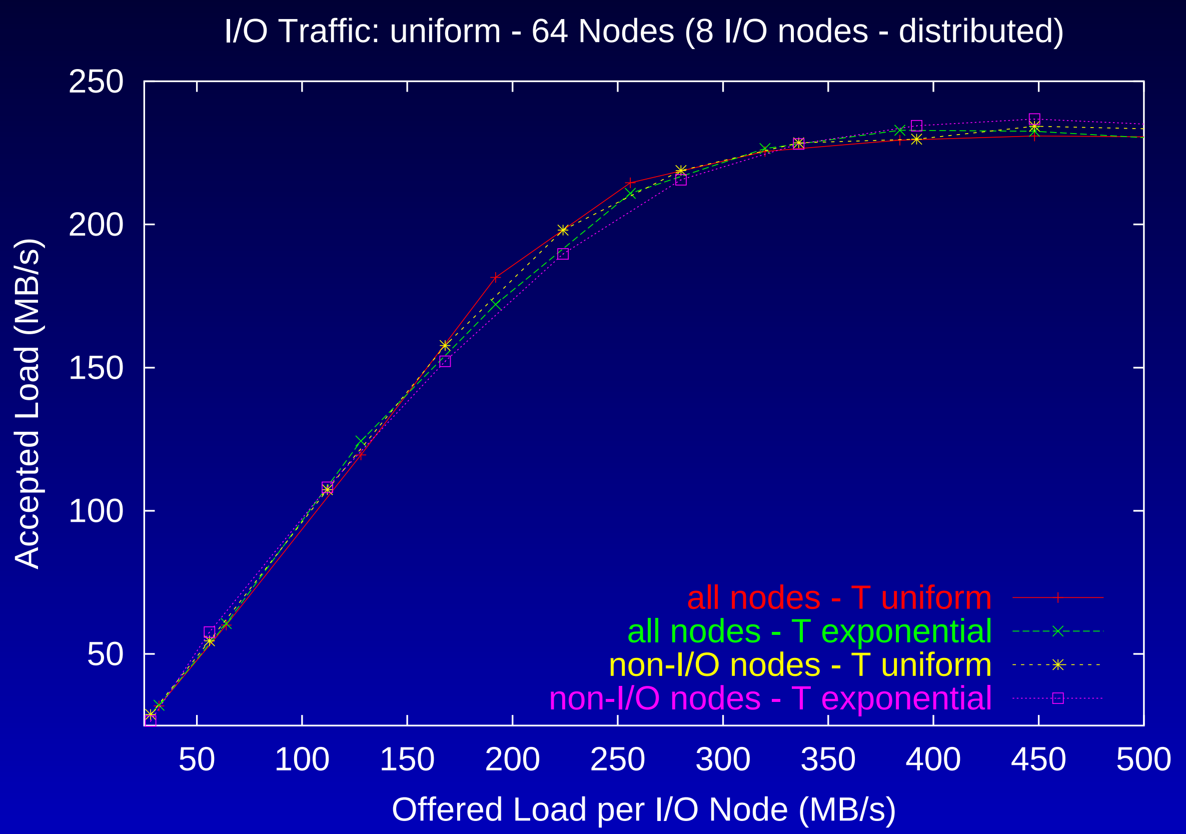


Placement of I/O nodes: Clustered Mapping



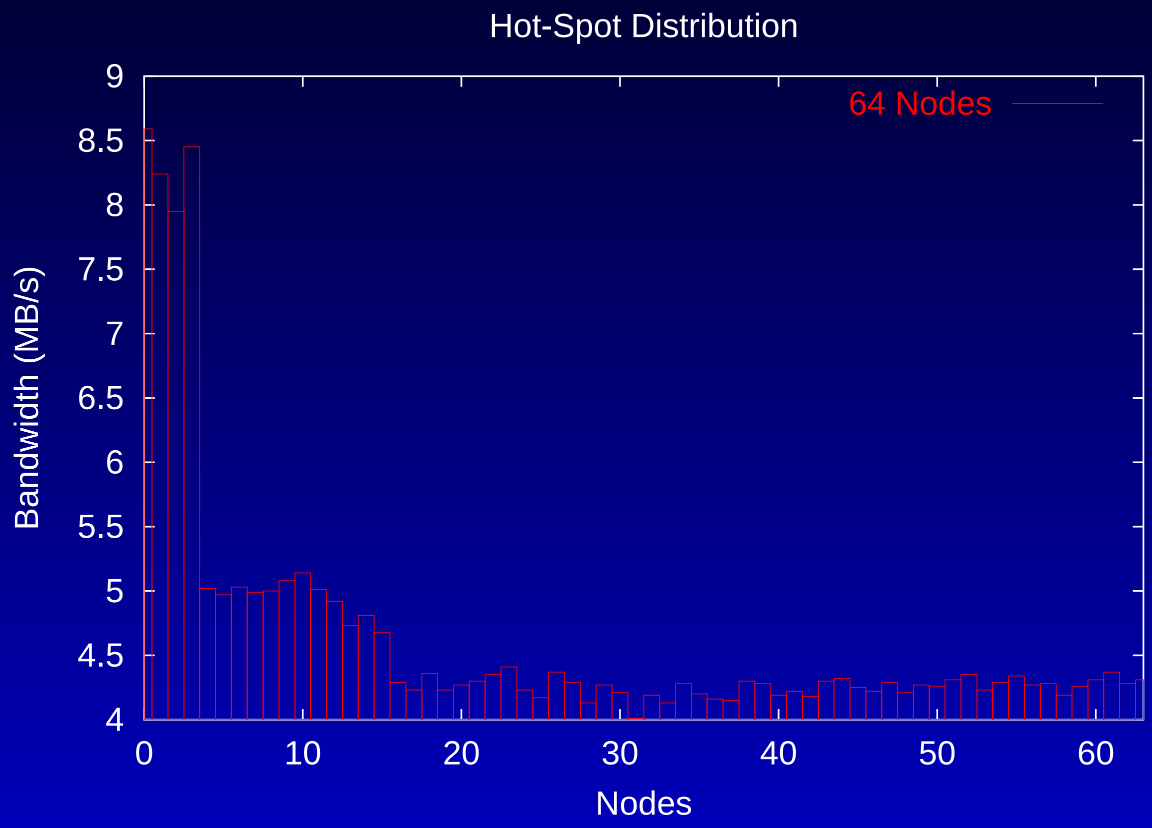
Bandwidth delivered by each I/O node.

Placement of I/O Nodes: Distributed Mapping



Bandwidth delivered by each I/O node.

Fairness Problem with Hot-Spot

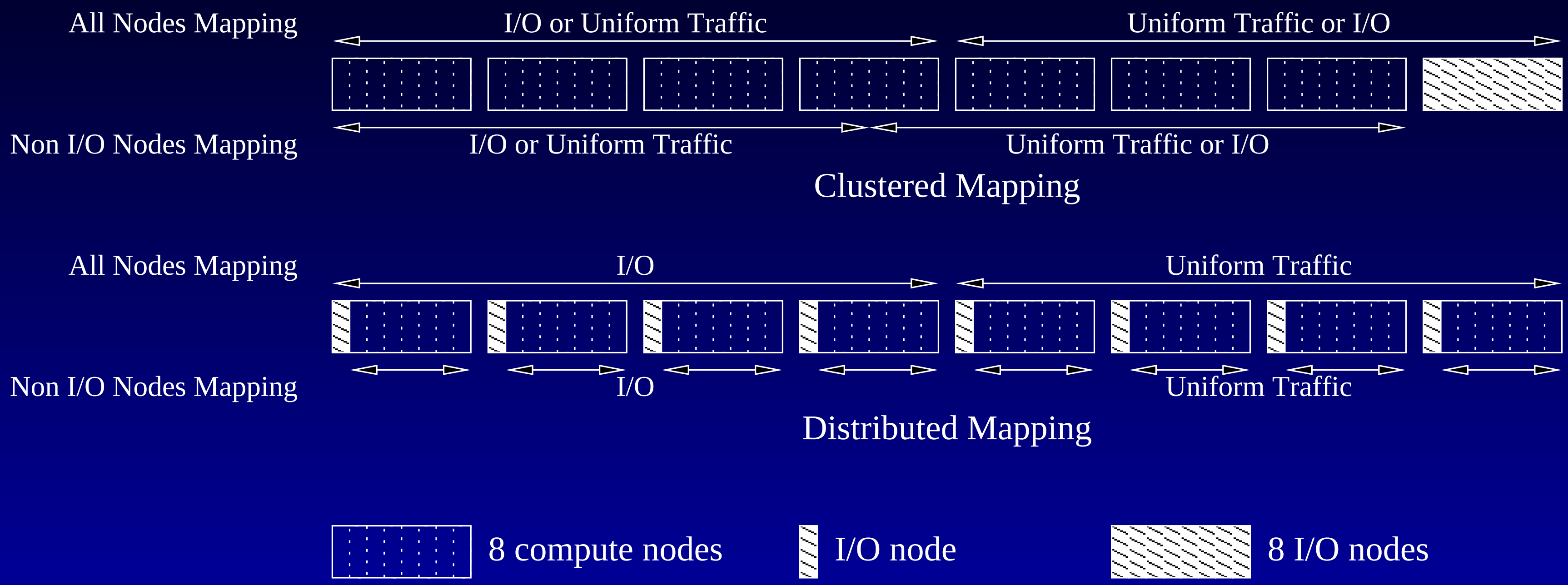


Bandwidth delivered to each node.

Performance results with a single parallel job

- Insensitive to read/write ratio.
- Insensitive to time and space distributions.
- Slightly sensitive to application mapping when the I/O nodes perform computation.
- Better performance with distributed mapping, due to a fairness problem in the network.

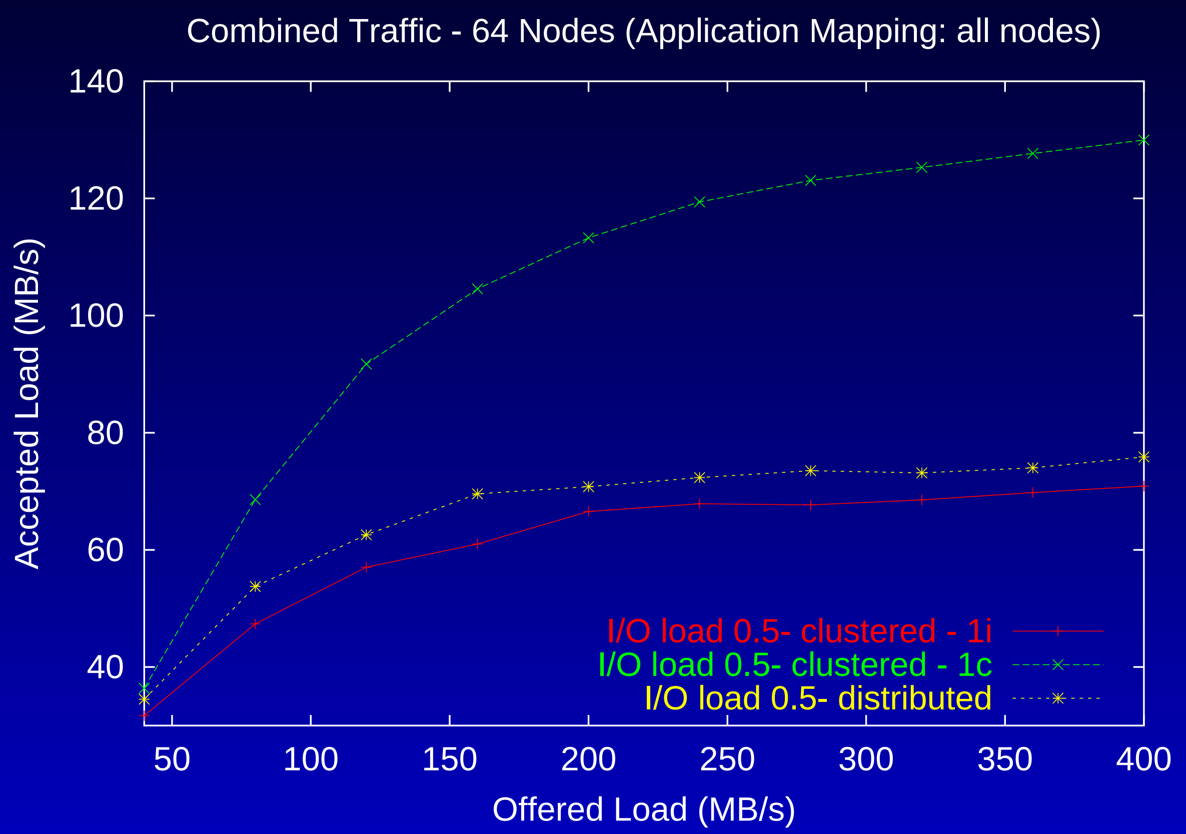
Interference of Background I/O Traffic



Performance analysis

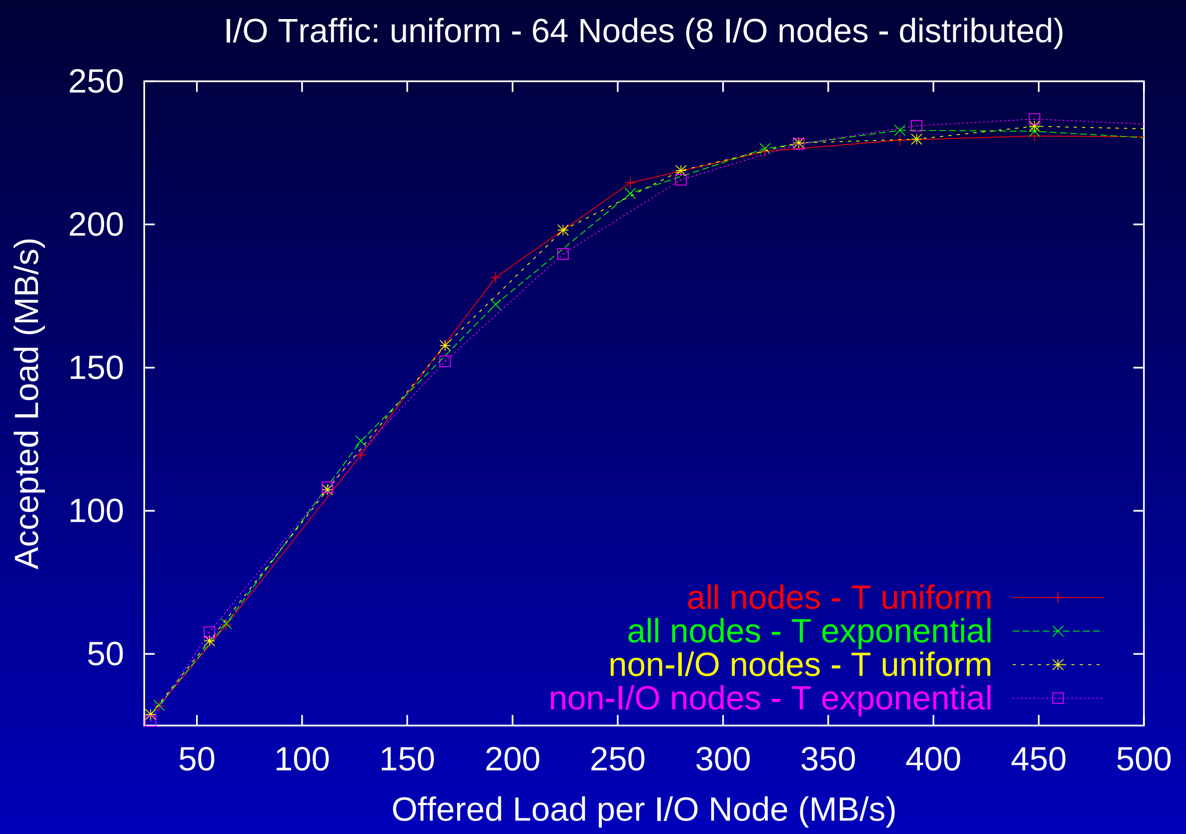
- We consider two distinct jobs, one doing I/O an another computation and communication.
- I/O node mapping (clustered and distributed).
- Application mapping (the I/O nodes can perform computation).

Application Mapping: All Nodes



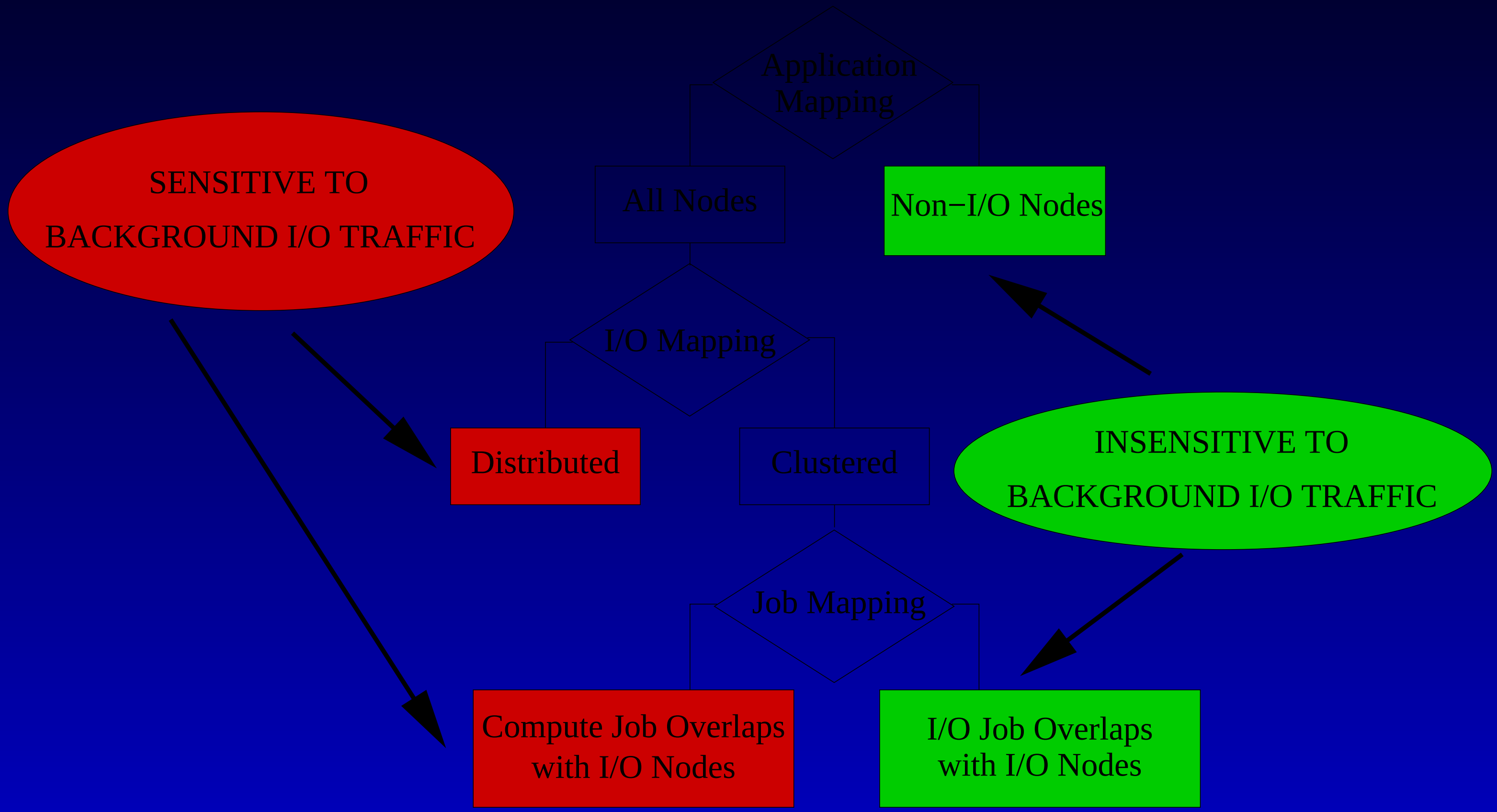
Bandwidth delivered by each compute node.

Application Mapping: non-I/O nodes



Bandwidth delivered by each compute node.

Interference of Background I/O Traffic



Placement of I/O Nodes in a Large Scale Cluster

Placement of I/O Nodes in a Large Scale Cluster

- With a single I/O job, it is better to distribute the I/O server rather that cluster them in a single segment of the network.

Los Alamos
NATIONAL LABORATORY

Placement of I/O Nodes in a Large Scale Cluster

- With a single I/O job, it is better to distribute the I/O server rather that cluster them in a single segment of the network.
- By doing so, we can get a bandwidth increase of about 20%

Placement of I/O Nodes in a Large Scale Cluster

- With a single I/O job, it is better to distribute the I/O server rather that cluster them in a single segment of the network.
- By doing so, we can get a bandwidth increase of about 20%
- The performance is insensitive to read/write ratio, mapping of the I/O job, inter-arrival time, access pattern.

Los Alamos NATIONAL LABORATORY

Placement of I/O Nodes in a Large Scale Cluster

- With a single I/O job, it is better to distribute the I/O server rather that cluster them in a single segment of the network.
- By doing so, we can get a bandwidth increase of about 20%
- The performance is insensitive to read/write ratio, mapping of the I/O job, inter-arrival time, access pattern.
- Multiple jobs can be run concurrently without interference, as long as these jobs are not mapped on the I/O nodes.

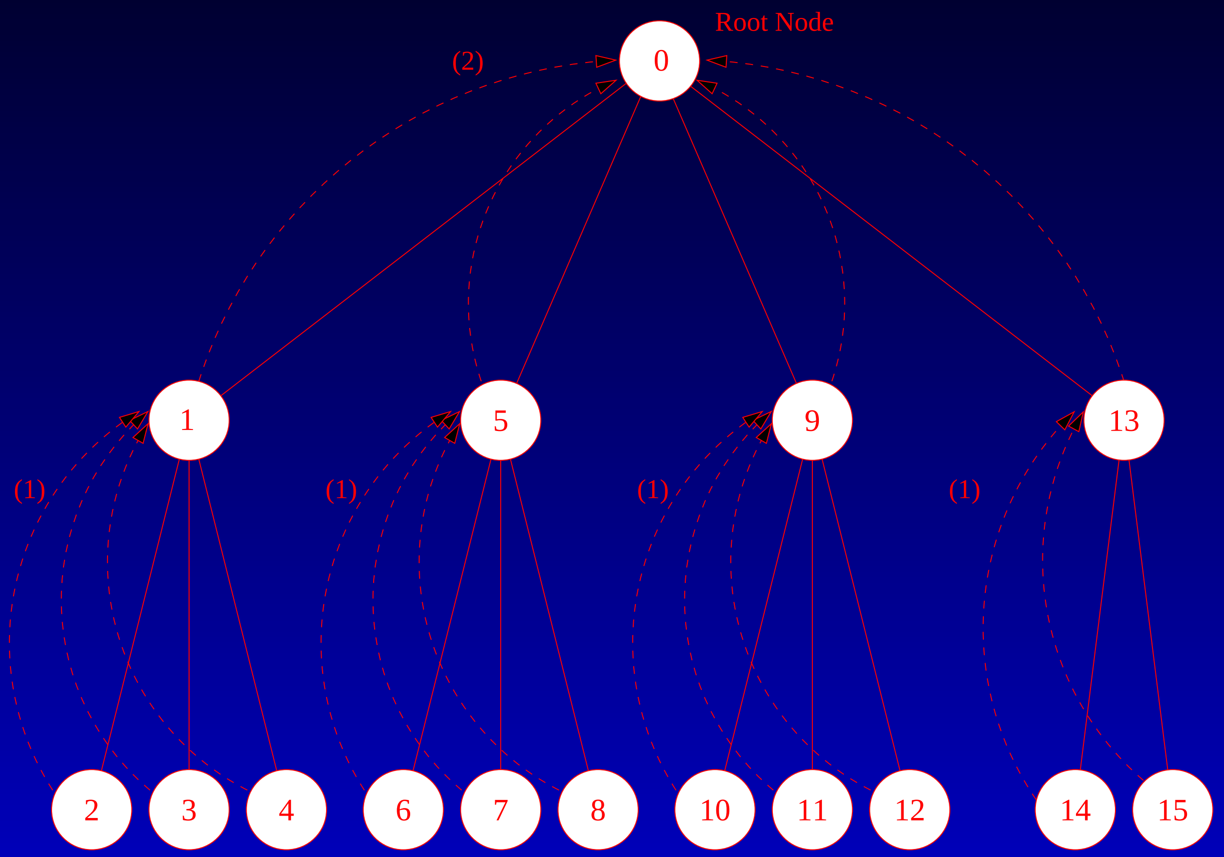
Los Alamos
NATIONAL LABORATORY

Collective Communication

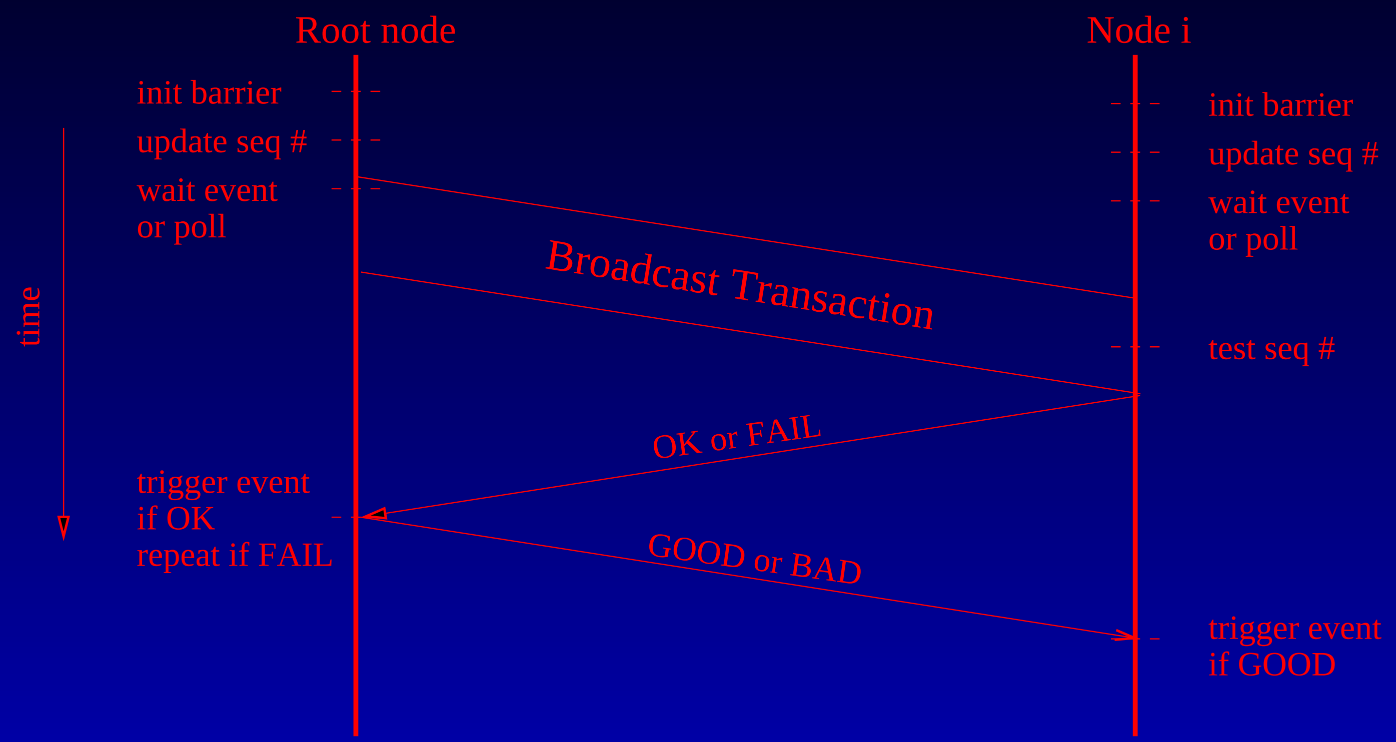
- The efficient implementation of collective communication is a challenging design effort.
- Very important to guarantee scalability of barrier synchronization, broadcast, gather, scatter, reduce etc.
- Essential to implement system primitives to enhance fault-tolerance.
- The experimental results are obtained on a 64-node Alphaserver cluster.
- In order to expose the real network performance, we place the communication buffers in Elan memory.
- Fabrizio Petrini, Salvador Coll, Eitan Frachtenberg and Adolfy Hoisie, *Hardware- and Software-Based Collective Communication on the Quadrics Network*, IEEE International Symposium on Network Computing and Applications 2001 (NCA 2001), Boston, MA.

Los Alamos
NATIONAL LABORATORY

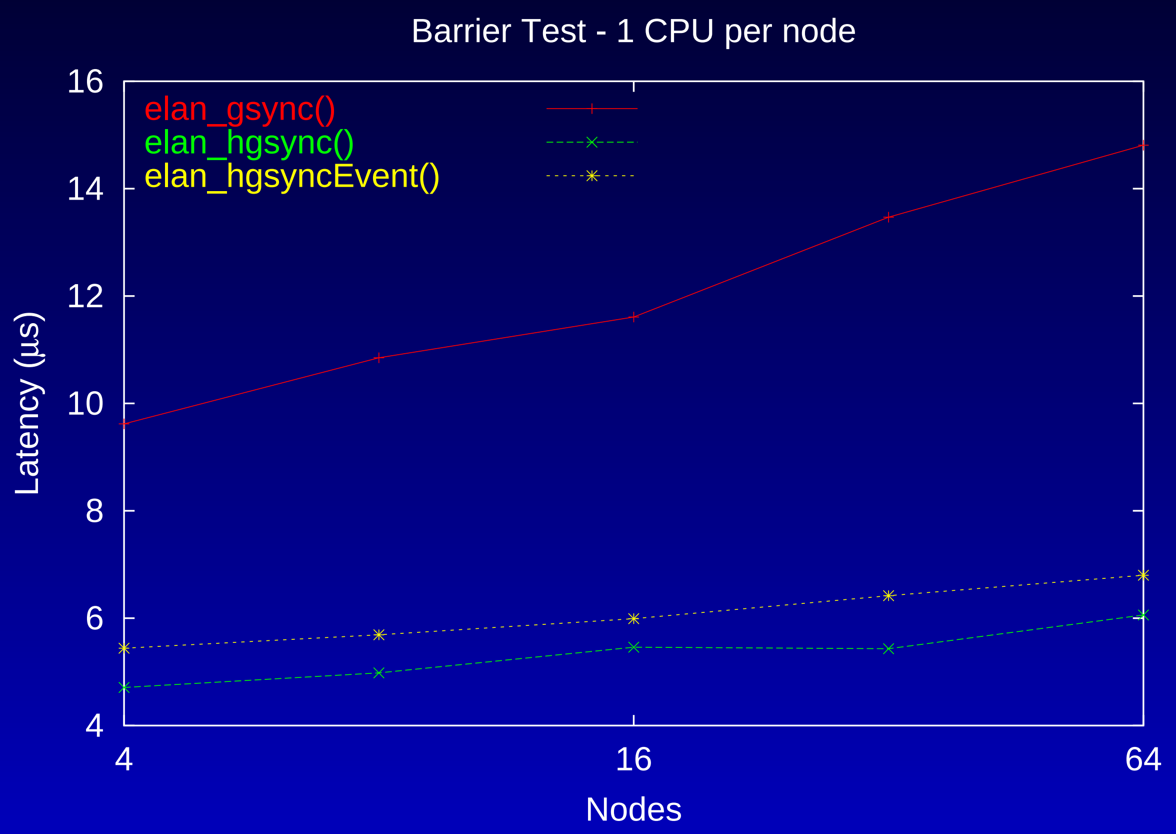
Software Multicast



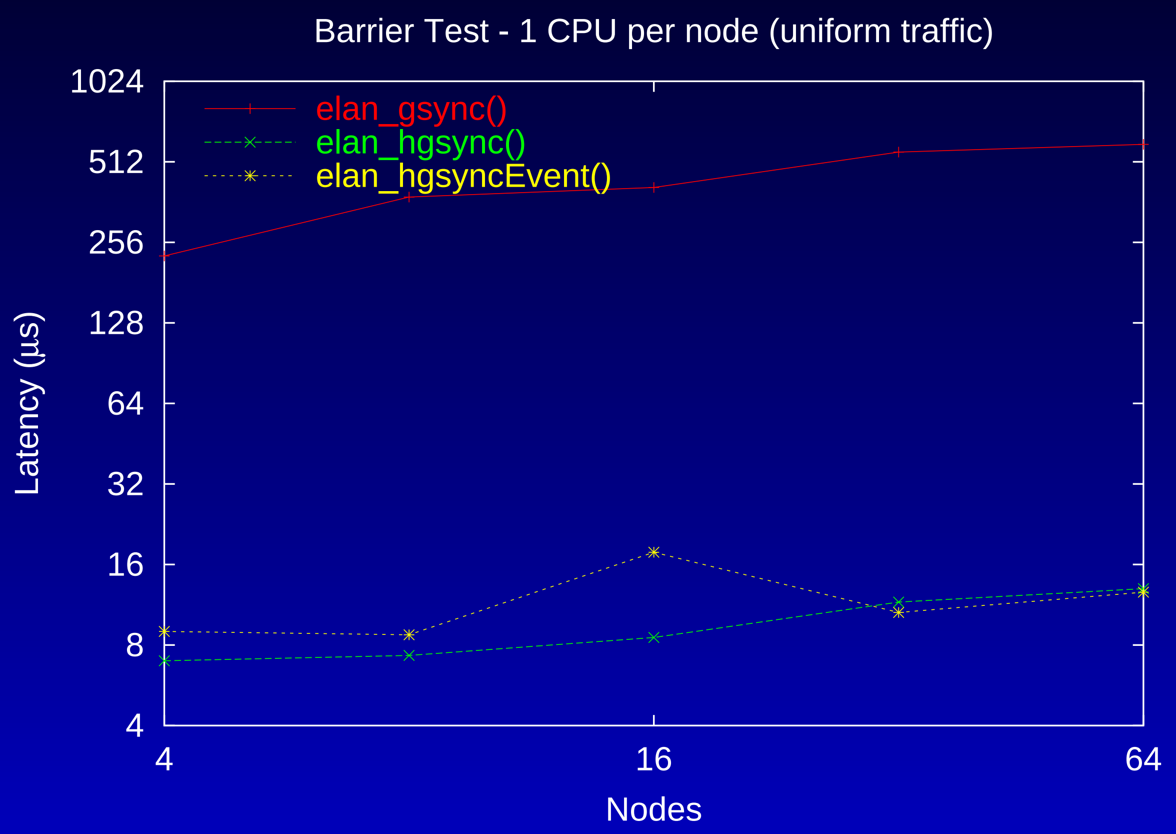
Hardware Multicast



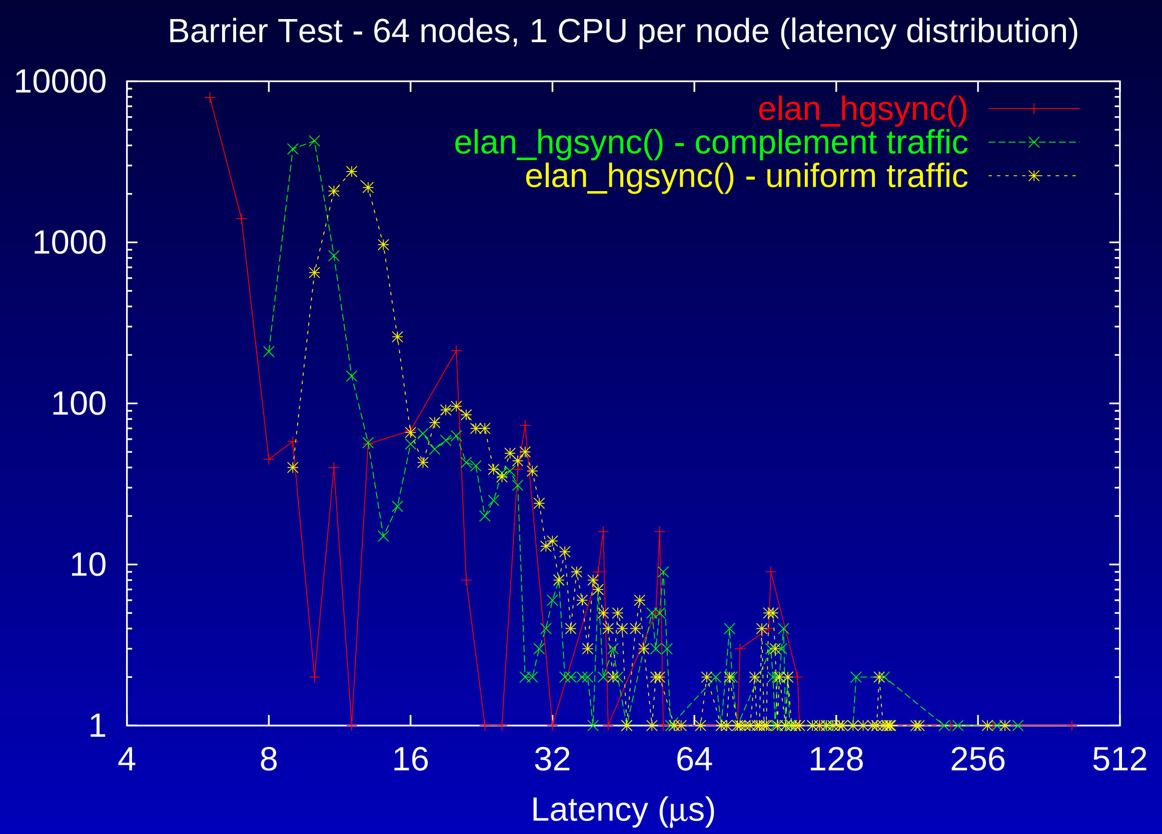
Barrier Synchronization



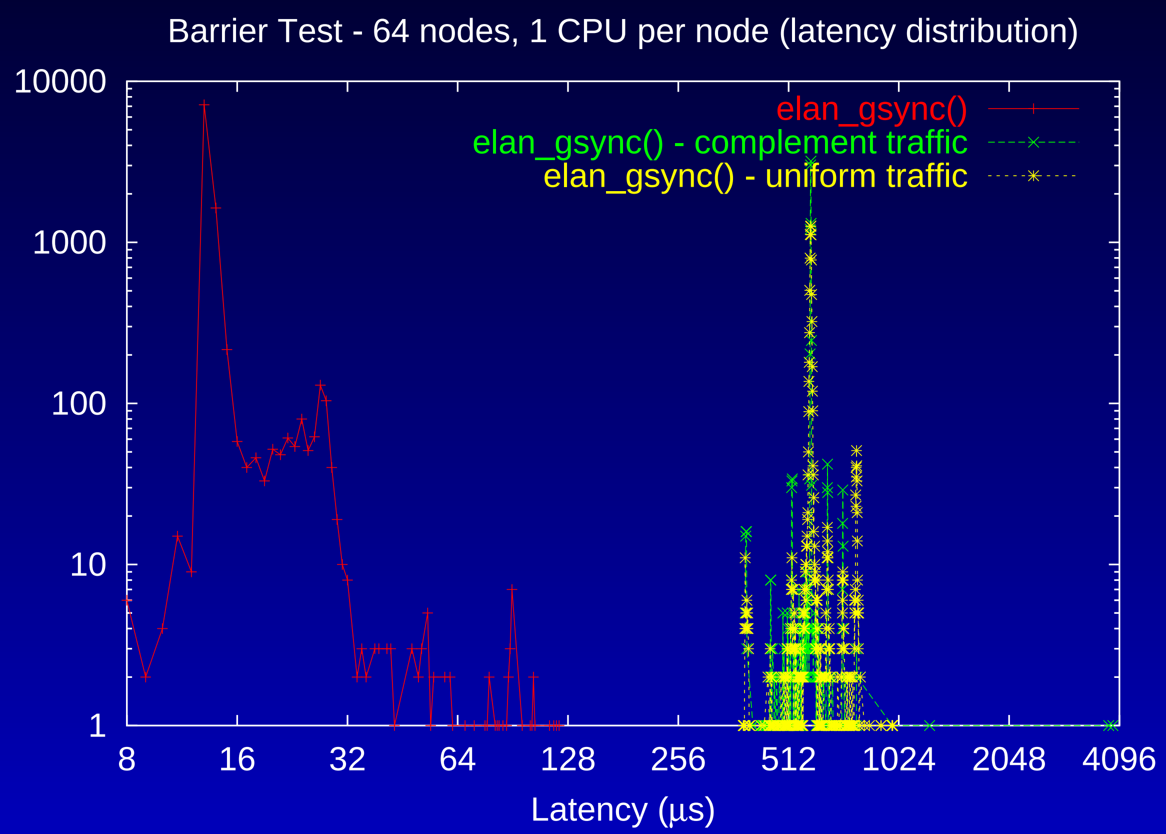
Barrier Synchronization with Background Traffic



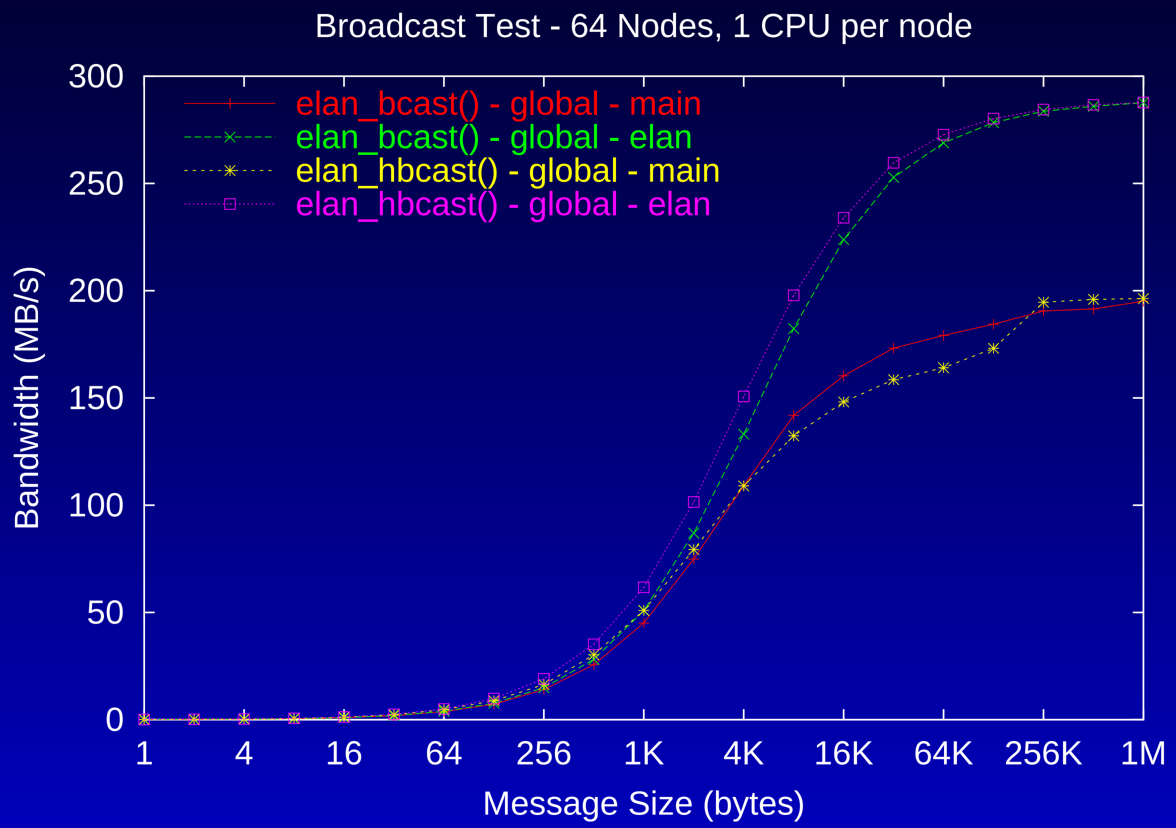
Hardware Barrier with Background Traffic



Software Barrier with Background Traffic

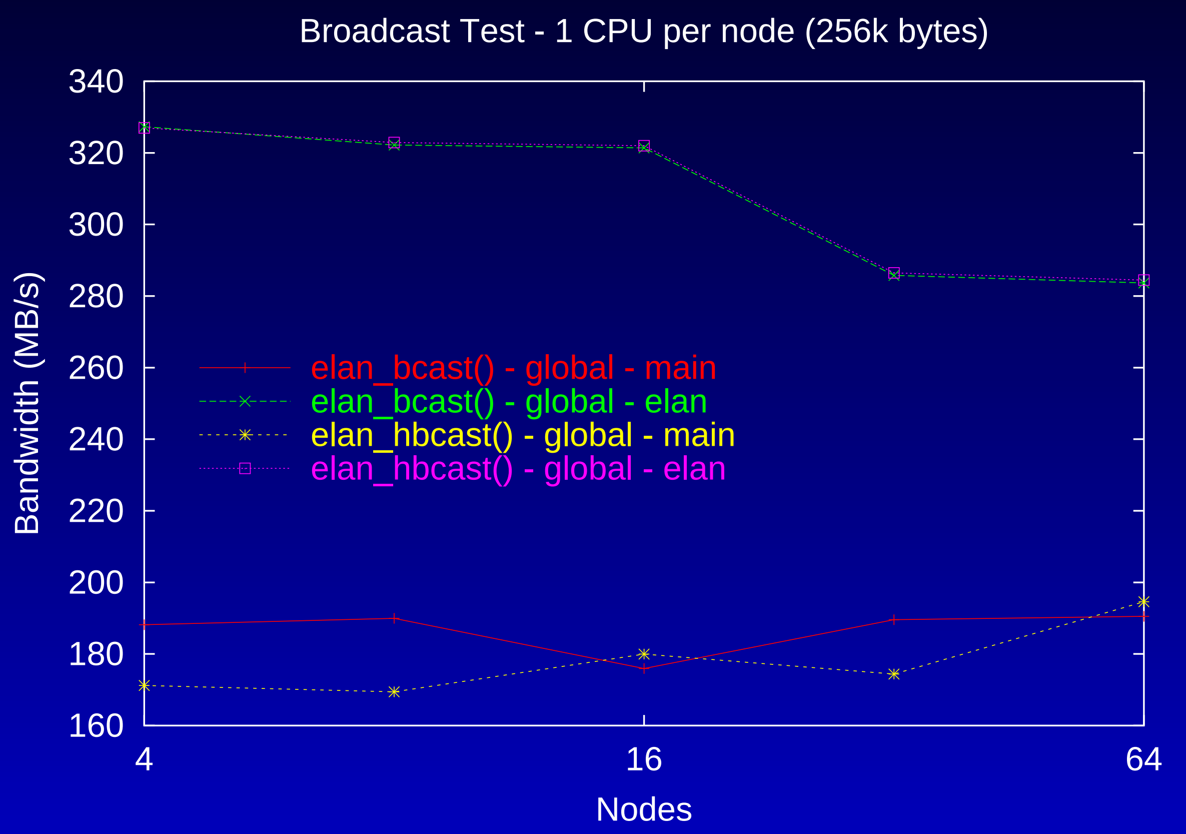


Broadcast



Los Alamos NATIONAL LABORATORY

Broadcast Scalability



Collective Communication

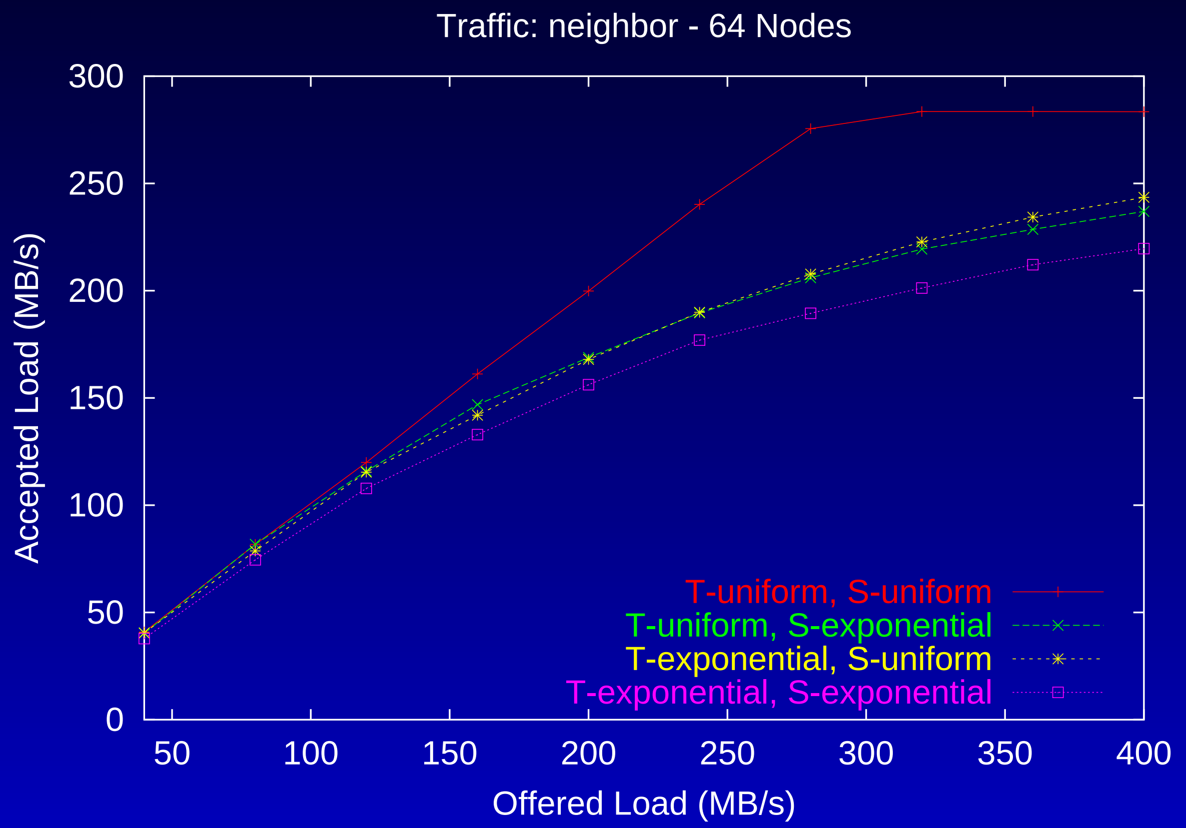
- Hardware-based synchronization takes as little as 6 μsec on a 64-node Alphaserver, with very good scalability.
- Good latency and scalability are achieved with the software-based synchronization too, which takes about 15 μsec.
- The hardware barrier is almost insensitive to background traffic, with 93% of the synchronizations delivered in less than 20 μsec.
- With the broadcast, both implementations can deliver a sustained bandwidth of 288 MB/sec Elan memory to Elan memory and 200 MB/sec main memory to main memory.

Permutation Patterns

In these permutation patterns each node sends to a single destination using a pre-defined permutation of the processing nodes. Not all the networks can handle these patterns efficiently.

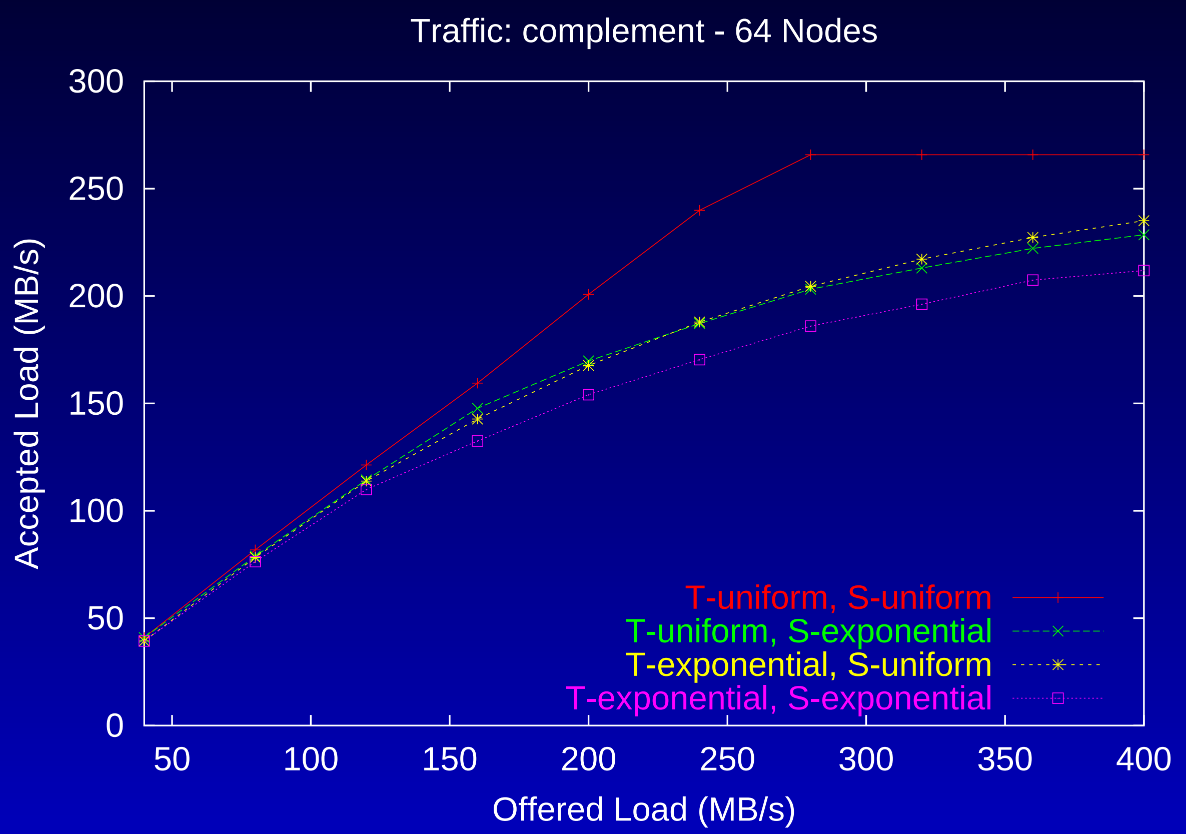
- Bit-reversal.
- Butterfly.
- Complement.
- Matrix transpose.
- Perfect-shuffle.
- Nearest neighbor.

Neighbor

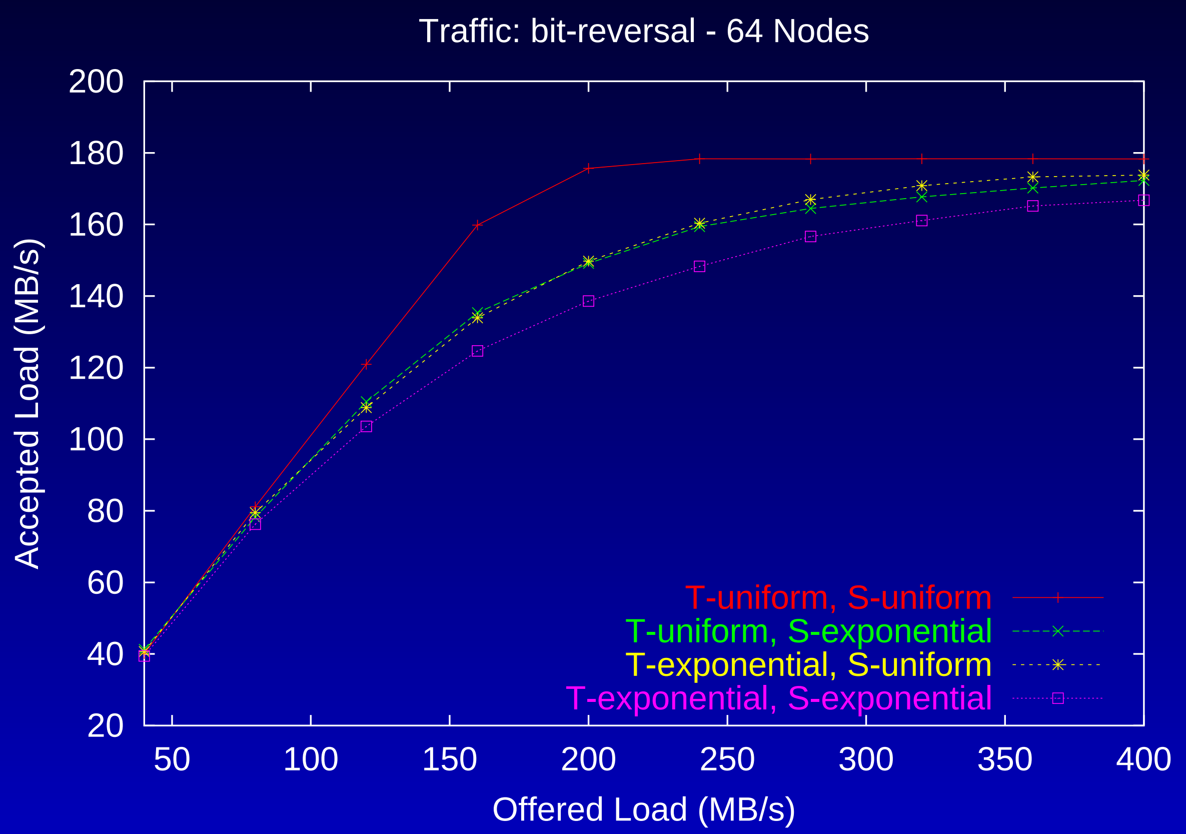


Traffic: neighbor - 64 Nodes
Accepted Load (MB/s)
Offered Load (MB/s)
0
50
100
150
200
250
300
50
100
150
200
250
300
350
400
T-uniform, S-uniform
T-uniform, S-exponential
T-exponential, S-uniform
T-exponential, S-exponential

Complement


Traffic: complement - 64 Nodes
300
250
200
150
100
50
0
Accepted Load (MB/s)
50
100
150
200
250
300
350
400
Offered Load (MB/s)
T-uniform, S-uniform
T-uniform, S-exponential
T-exponential, S-uniform
T-exponential, S-exponential

Bit Reversal



Congestion Matrix for Complement Traffic

0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
58 58 58 58	58 58 58 58	58 58 58 58	58 58 58 58	58 58 58 58	58 58 58 58	58 58 58 58	58 58 58 58	58 58 58 58	58 58 58 58	58 58 58 58	58 58 58 58	58 58 58 58	58 58 58 58	58 58 58 58	58 58 58 58

79 79 79 79	79 79 79 79	79 79 79 79	79 79 79 79	79 79 79 79	79 79 79 79	79 79 79 79	79 79 79 79	79 79 79 79	79 79 79 79	79 79 79 79	79 79 79 79	79 79 79 79	79 79 79 79	79 79 79 79	79 79 79 79
100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
37 37 37 37	37 37 37 37	37 37 37 37	37 37 37 37	37 37 37 37	37 37 37 37	37 37 37 37	37 37 37 37	37 37 37 37	37 37 37 37	38 37 37 37	37 37 37 37	37 37 37 37	37 37 37 37	37 37 37 37	37 37 37 37

100 100 99 99	100 100 100 100	99 100 100 100	99 99 99 99	100 100 100 100	100 100 100 100	99 99 99 99	100 100 100 100	100 100 100 100	99 99 100 99	100 100 100 100	100 100 100 100	100 100 100 100	100 100 100 100	100 100 100 100	100 100 100 100
100	100	100	99	100	100	100	100	100	100	100	100	100	100	100	100
17 17 17 17	17 17 17 17	17 17 17 17	17 17 17 17	17 17 17 17	17 17 17 17	17 17 17 17	17 17 17 17	17 17 17 17	17 16 17 17	17 17 17 17	17 17 17 17	17 16 17 17	17 17 17 17	17 17 17 17	16 17 17 17

<=0% <=20% <=40% <=60% <=80% <=100%

Congestion Matrix for Bit Reversal Traffic

0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
17	19	18	18	17	19	17	17	16	19	18	17	17	19	18	18
14 13 11 12	13 13 14 16	12 14 14 13	13 13 14 11	14 13 11 13	14 13 14 16	12 14 13 12	13 13 14 12	13 12 10 12	13 13 14 16	12 14 14 13	13 13 13 12	14 13 11 12	13 13 14 16	12 14 14 12	14 13 14 12

21 20 20 20	42 41 44 43	48 44 47 45	30 30 29 31	37 39 34 38	43 46 45 42	26 23 23 25	36 39 37 39	47 46 45 45	36 35 36 37	26 28 29 26	26 26 27 28	41 37 38 39	20 20 19 20	39 40 37 40	49 47 46 47
33	68	76	50	62	73	41	62	76	59	45	44	65	33	64	78
1 9 6 1	1 10 20 1	1 20 19 1	1 14 11 1	1 11 23 1	1 21 15 1	1 8 13 1	1 13 18 1	1 22 17 1	1 13 17 1	1 7 13 1	1 9 10 1	1 16 21 1	1 8 7 1	1 16 15 1	1 20 21 1

17 26 26 20	24 100 89 46	38 50 84 61	17 25 24 20	21 25 18 21	92 72 50 79	48 92 32 60	22 25 17 22	26 22 19 20	77 70 57 43	98 60 34 44	24 22 21 20	21 16 24 25	87 29 65 92	62 33 76 89	22 17 23 24
31	88	79	30	30	100	80	30	30	85	80	30	30	93	89	30
0 1 1 1	1 1 1 1	1 1 1 1	0 1 1 1	1 1 0 1	1 1 1 1	1 1 1 1	1 1 0 1	1 0 1 1	1 1 1 1	1 1 1 1	1 0 1 1	1 1 1 0	1 1 1 1	1 1 1 1	1 1 1 0

<=0% <=20% <=40% <=60% <=80% <=100%

Hardware- and software-based collectives

- Important results to enhance the fault-tolerance of a large-scale machine.
- Fabrizio Petrini, Salvador Coll, Eitan Frachtenberg and Adolfy Hoisie. Hardware- and Software-Based Collective Communication on the Quadrics Network. In IEEE International Symposium on Network Computing and Applications 2001 (NCA 2001), Boston, MA, October 2001.

Job Scheduling and Resource Management

- Study of the scalability of RMS and gang scheduling
- Eitan Frachtenberg, Fabrizio Petrini, Salvador Coll and Wu-chun Feng. Gang Scheduling with Lightweight User-Level Communication . In 2001 International Conference on Parallel Processing (ICPP2001), Workshop on Scheduling and Resource Management for Cluster Computing, Valencia Spain, September 2001.

Resources

More information on our work can be found at

- www.c3.lanl.gov/~fabrizio/quadrics.html